094103

PRS-ESA-85-007

4 February 1985
14

19981022 079

East Europe Report

SCIENCE AND TECHNOLOGY

FBIS FOREIGN BROADCAST INFORMATION SERVICE

6
121
A06

NOTE

JPRS publications contain information primarily from foreign newspapers, periodicals and books, but also from news agency transmissions and broadcasts. Materials from foreign-language sources are translated; those from English-language sources are transcribed or reprinted, with the original phrasing and other characteristics retained.

Headlines, editorial reports, and material enclosed in brackets [] are supplied by JPRS. Processing indicators such as [Text] or [Excerpt] in the first line of each item, or following the last line of a brief, indicate how the original information was processed. Where no processing indicator is given, the information was summarized or extracted.

Unfamiliar names rendered phonetically or transliterated are enclosed in parentheses. Words or names preceded by a question mark and enclosed in parentheses were not clear in the original but have been supplied as appropriate in context. Other unattributed parenthetical notes within the body of an item originate with the source. Times within items are as given by source.

The contents of this publication in no way represent the policies, views or attitudes of the U.S. Government.

PROCUREMENT OF PUBLICATIONS

JPRS publications may be ordered from the National Technical Information Service, Springfield, Virginia 22161. In ordering, it is recommended that the JPRS number, title, date and author, if applicable, of publication be cited.

Current JPRS publications are announced in Government Reports Announcements issued semi-monthly by the National Technical Information Service, and are listed in the Monthly Catalog of U.S. Government Publications issued by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

Correspondence pertaining to matters other than procurement may be addressed to Joint Publications Research Service, 1000 North Glebe Road, Arlington, Virginia 22201.

JPRS-ESA-85-007

14 February 1985

EAST EUROPE REPORT

SCIENCE AND TECHNOLOGY

CONTENTS

GERMAN DEMOCRATIC REPUBLIC

OPERATING SYSTEM OC 7 EC FOR ESER 1, 2 COMPUTERS

System Development, Applications Described

East Berlin EDV ASPEKTE in German No 3, 1984 (signed to press 30 May 84) p 1

[Article by Walter Muench, editorial staff member, VEB ROBOTRON Center for Research and Engineering]

[Text] The complexity and scope of information processing that must be dealt with in the economy is steadily increasing and is imposing ever more stringent requirements on the performance capability of information processing systems. This makes it necessary to develop further the device technology and program support that will correspond to these requirements.

Within the framework of many years of close collaboration in ESER (uniform data processing system), especially with collectives of the USSR, an important performance stage could be essentially concluded in operating system development on the part of the VEB Combine Robotron with the collaboration of the VEB Combine Data Processing. Industrial testing could be begun.

This issue is concerned with EDP aspects and gives a detailed overview concerning the operating system complex OC-7 EC including components that can be delivered separately. The operating system complex OC-7 EC comprises the operating systems SVM, BPS, and SVS. The starting basis of development for BPS and SVS was the eighth modification of the OC-6.1 EC, for SVM it was edition 1 of the SVM/ES. The operating systems SVM, BPS, and SVS can run only on models of Series 2 of ESER. Series 2 also includes, among others, the GDR models EC 1055, EC 1055M, and EC 1056.

The SVM is an operating system which implements the concept of virtual machines. ESER operating systems can function on these virtual machines. In PTS, SVM has a powerful component for dialogue operation.

The BPS is a batch processing system which can run only on a virtual machine of the SVM. Its use is therefore possible only in conjunction with the SVM. The component PTS of the SVM is available for dialogue operation.

Within the framework of the development of the OC-7 EC, new models and devices of ESER are supported within the system; significant functional expansions are implemented - for example multi-machine support and remote processing. In

particular, it is possible to achieve a significant increase of program throughput.

The fast and wide introduction of the OC-7 EC in practical computing operations therefore is of special interest to the national economy. The OC-7 EC will, in coming years, become very widespread among users of models of Series 2 of ESER. Further work on the OC-7 EC will start from the idea that future applications of the OC-6.1 EC will concentrate on models of Series 1 of ESER. New operating system functions, which are relevant for Series 2, will therefore essentially be furnished within OC-7 EC.

The individual papers in this issue choose their presentations in such a fashion that newly developed functions are explained mainly with reference to the initial systems. Furthermore, at appropriate points, data are given concerning efficiency increases. It is here supposed that the reader is familiar with the basic ideas of the ESER operating system.

The last pages of this issue treat technologies and programs which can be used in developing program complexes. They are thus to a certain extent related with the operating system complex that has been presented.

Development of OS/ES System

East Berlin EDV ASPEKTE in German No 3, 1984 (signed to press 30 May 84) pp 2-4

[Article by Walter Muench, editorial staff member, VEB ROBOTRON, Center for Research and Engineering]

[Text] 1. Development of the OS/ES to the OC-6.1 EC

The OS/ES operating system was designed for medium and high-power models of the ESER. Compared to the DOS/ES, which is tailored for operating low-power models, it offers greater convenience, more automation, and an extended functional scope.

Including Edition 4, which became operational in 1976, the OS/ES is an operating system which is intended to support the functional and hardware scope of Series 1 of the ESER, which includes the model EC 1040. This edition contains the control program configuration

- PCP (monoprogram operation)[1]
- MFT (multiprogram operation with a fixed number of jobs) and
- MVT (multiprogram operation with a variable number of jobs).

Initially, only batch operation is implemented. Subscriber operation will be supported in the next stage by the dialogue entry component (CRJE) for locally

---

[1] Remark: The abbreviations used in the various papers are explained in our index of abbreviations on page 19. The Editors.

connected devices. To support remote data processing, the BTAM access method is made available. Besides the ASSEMBLER, the language translators for RPG, COBOL, FORTRAN IV, ALGOL, and PL/1 are offered. In the BC mode, Edition 4 can indeed run on models of Series 2, but does not use or support the new functions of this series. The functional scope of Series 2 of ESER, which includes the models EC 1055 and EC 1055 M, is supported for the first time in Edition 6.1 of the OS/ES by the system configurations SVS (system with a virtual address space). This edition represents a new functional level of the OS/ES development. It was made available in its initial version in 1979. Because of its special significance, edition 6.1 is also called OC-6.1 EC.

The OC-6.1 EC contains the control program configurations MFT, MVT, and SVS. It is thus an operating system for Series 1 and 2 of ESER. Only MFT and MVT can be operated on models of Series 1; MFT and MVT can be operated on models of Series 2 in the BC mode, and SVS can be operated in the EC mode.

With SVS, multiprogram operations of level 64 is possible in a virtual memory with a capacity up to 16 mbytes.

Modification 4 of the OC-6.1 EC was delivered beginning in 1981.

The most important functional extensions are:

- Subscriber support (TSO) for MVT;
- expanded remote data processing access methods TCAM;
- generalized access methods for direct access units VSAM;
- PL/1 optimization compilor and PL/1 test compiler.

The modification 8 of OC-6.1 EC appeared in 1982. It is distinguished by the following improvements:

- Increasing the functional scope, among other things by the programming systems FORTRAN, COBOL, and PL/1. The programming systems comprise at least one translator and one running time system.
- Increasing the power, among other things by the integrated SPOOL system SPAM, which was made available in its first version, and the expanded SORTING/MIXING program (for MFT, MVS, and SVS on models of Series 2).
- Supporting new I/O units, among others the floppy disk unit technology and the matrix module of the central unit (CPU) EC 2655 and EC 2655 M.
- System linkage between SVS and SVM/ES. In this way, effective operation of SVS on a virtual machine under SVM/ES is facilitated.
- Increasing data protection.

The distribution system of the eighth modification contains about 3.3 million assembler statements.

The programming systems PL/1, FORTRAN, and COBOL, as well as RPG, SORTING/MIXING expanded, TSO and Assembler 2, are furnished as components of Type 2, in addition to the distribution system. Components which are separately marketable are designated as components of Type 2. They receive their own output and modification designations.

The overall scope of Type 2 components which can run with the eighth modification comprises about 4.2 million assembler statements.

The development of the OC-6.1 EC was essentially concluded with the ninth modification, which has been used since the fourth quarter of 1983. Further work will concern the support of new equipment, troubleshooting, and to a limited extent the achievement of runnability of new or further developed components of Type 2. For this purpose, alteration packets to the distribution system will be published at certain time intervals. In 1983, a PASCAL programming system has been added to the components of Type 2, and TSO will also be furnished for the control program configuration SVS. Beginning in the fourth quarter of 1984, the multimachine support POC will be available for MVT and SVS as a component of Type 2. This will implement the priniciples of host computer/slave computer. The further application of the OC-6.1 EC will increasingly concentrate on models of Series 1.

2. Operating-System Support of Series 2 of the ESER

As the Series 2 models of ESER are becoming more widespread in application, it became suitable to furnish operating systems which in particular make attractive the functional capabilities of these models. Within the framework of the ESER, the following are offered as typical operating system of the ESER Series 2, which can no longer run on models of Series 1:

- DOC-3 EC with its further developments, designed for low-power models of Series 2
- SVM/ES
- OS/ES, Edition 7

The SVM/ES operating system implements the concept of virtual machines. The virtual machine represents an extension of the concept of virtual memories in the sense that all elements of a data processing system (DP system) are implemented virtually. The number of virtual machines that can be set up on a real machine and that can run at the same time is practically limited only by the power of the configuration of the DP system.

The availability of the SVM/ES offers to the user the following new or improved capabilities:

- Parallel use of different operating systems in different virtual machines. This brings with it significant advantages when changing the operating system, when making a transition to a new operating system, and in the test phase.
- Parallel processing of mutually independent problems on various virtual machines of one DP system, especially parallelism of batch operation and dialogue operation, of program development jobs, and of technical maintenance for productive operation.
- Integration of remote data processing into the overall system by remote virtual operating units as well as by the availability of RFTS and PTS.

- High processing reliability and data protection through the work of various users in mutually independent virtual machines.

However, when operating an ESER operating system under SVM/ES, there is a reduction of program throughput. Edition 1 of the SVM/ES has been delivered since 1982. This edition can run effectively only on the models EC 1055 and EC 1055 M, since only in the CPU EC 2655 and EC 2655 M is an appropriate microprogram support implemented.

3. OC-7 EC

Starting from the eighth modification of the OC-6.1 EC and Edition 1 of the SVM/ES, and taking into account the user requirements for the development of the OS/ES, Edition 7, the following objectives were set up:

- support of medium and high-power models of Series 2 of the ESER
- significant increase of the program throughput
- functional expansion, among other things, support of remote processing by means of a locally remote data processing units and multiprocessor configurations
- expanded application of the principle of virtual machines within the system
- support of new I/O units and external memory
- furnishing further developed or newly developed components of Type 2.

The OS/ES, Edition 7 no longer supports the models of Series 1 of ESER and can no longer run on the models of Series 1. The control program configurations MFT and MVT are therefore no longer contained in this edition. Since Edition 7, as compared to Edition 6.1, represents a new functional level, the term OC-7 EC is also applied to it.

The OC-7 EC actually represents an operating system complex which at this time contains the operating systems SVM, BPS and SVS.

The above-mentioned operating systems are marketed separately in their basic features and have their own edition designations. The SVM operating system currently lies at the level of Edition 3.0 (SVM 3.0). SVM 3.0 represents a further development of the SVM/ES. It is primarily characterized by the following improvements, which are effected in the CP and PTS:

- support of new models and units of the ESER, Series 2, including support of the matrix module of EC 2655/EC 2655 M/EC 2156 in the CP and PTS
- expansion of the functional scope, including support of symmetric and asymmetric multiprocessor complexes as well as the VSAM and VSAM utility program AMS in the PTS
- improvement of the utilization of real resources, including the capability of dynamic transfer of the work of one operating system on one virtual machine to working on a real machine and vice versa
- improvement of file compatability with respect to BPS and SVS
- expansion of means for dialogue operation
- increased operating convenience
- increased data protection.

The SVM 3.0 comprises about 0.6 million assembler statements. To guarantee the effective operation of SVM 3.0, powerful microprogram supports are contained in the CPU EC 2655, EC 2655 M, and EC 2156.

The BPS and SVS operating systems represent a direct further development of the SVS of the eighth modification of the OC-6.1 EC. With respect to this status of the SVS, they are upwards compatible to the problem programming levels. SVM, BPS, and SVS are compatible on the levels of source text, object modules, and loader modules between BPS and SVS. They have a high degree of file compatability.

The BPS operating system is a batch processing system which can run only on a virtual machine of the SVM. Dialogue operation is then implemented through the PTS of the SVM.

Speaking more precisely, the BPS is not an independent operating system, since it can be used only in conjunction with the SVM. The BPS makes possible the multiprogram operation from one level up to 15 per virtual machine. A multiple virtual memory up to 16 Mbytes in batch operation and dialogue operation is supported through the SVM. Compared to the SVS of the OC-6.1 EC, the functional scope in the BPS was expanded, but in some cases was also reduced.

With the BPS, as compared to the SVS of the eighth modification of the OC-6.1 EC, there is a noticeable increase of program throughput. With a measurement job stream, under certain conditions, a throughput increase of up to 45 percent was measured on the EC 1055 M. The BPS is available in status BPS 7.0. The distribution system contains about 1.8 million assembler statements. The operating system SVS can run on the real machines of the ESER, Series 2, as well as under control of the SVM for batch operation on virtual machines (V = V) or on virtual machines with reserved real memory (V = R). It implements batch operation and dialogue operation. The degree of multiprogram operation is 64. Compared to the SVS of the OC-6.1 EC, it is distinguished by the following improvements:

- A significant increase of program throughput by shortening the control program running sequences and further efficiency-enhancing measures. With a measurement job stream, a throughput increase of about 100 percent was measured on the EC 1055 M, under certain conditions, as compared to the SVS of the eighth modification of the OC-6.1 EC.
- Functional expansion
- Support of new models and units.

The SVS is available in status 7.0. The distribution system contains about 2.5 million assembler statements.

In connection with improving system linkage of the SVS to the SVM, measurements on the EC 1055 M have shown the following: Compared to operation of the SVS of the OC-6.1 EC, on a real machine which operates with SVS 7.0 under SVM 3.0,

- for V=V machines, the throughput can be increased 20 percent
- for V=R machines, the throughput can be increased 55 percent.

Comparative efficiency measurements between the SVS 7.0 and the DOC-3 EC on the EC 1055/EC 1055 M have not been performed as of this time. Because of the different purposes of these operating systems and because of the existing incompatabilities, such measurements would be complicated. In his article, "DOS-3 ES Operating System" /1/, Holenda specified that the DOC-3 EC proved to be "50 to 70 percent faster than OC-6 EC". This statement can at best refer to the eighth modification of the OC-6.1 EC.

Taking into account the efficiency increases explained above, when using the SVS 7.0 as compared to the SVS of the eighth modification of the OC-6.1 EC, there accordingly surely are no efficiency advantages of the DOC-3 EC as compared to the SVS 7.0.

For BPS and SVS in particular, the availability of the following new functions should be emphasized:

- VSAM2 with expanded functional scope, whereby the application areas are significantly widened and especially work with data bases is improved.
- A new remote data processing access method TCAM/NF. This access method supports hierarchical network structures which are connected via multiplexors or equivalent remote data processors in the emulation mode or via local remote data processors in the network control mode, with the central data processing system.

The function CRJE is no longer contained in the operating systems BPS and SVS of the OC-7 EC. Within the framework of the OC-7 EC, users of the SVS can choose between the much more powerful solutions TSO or PTS of the SVM for the implementation of dialogue operation, according to their job profiles.

Furthermore, the function DDM (dynamic test monitor) and the compiler ALGOL are no longer offered.

The following components of Type 2 are available for the OC-7 EC, as a new development or further development:

- the COBOL programming system, Edition 1.2
- the FORTRAN programming system and the FORTRAN OE compiler, Edition 1.1
- the PASCAL programming system, Edition 2.0
- the PL/1 programming system, Edition 2.2
- ASSEMBLER 2, Edition 2.1
- Multicomputer support POC, Edition 3.1
- TSO (only for SVS), Edition 2.0
- SORTING/MIXING 2, Edition 1.4 and
- RPG 2, Edition 1.0.

All the above-mentioned programming systems contain translators, by means of which dialogue operation is possible both in the TSO of the SVS and in the PTS of the SVM. With the FORTRAN OE and PL/1-OC translators of the PL/1 programming systems, optimizing compilers are becoming available.

The above-mentioned further developed components of Type 2 make it possible to delete the components assembler and translator for PL/1, COBOL and FORTRAN of the OC-6.1 EC in the OC-7 EC.

SVS 7.0 and SVM 3.0 will become available in the GDR beginning in 1985. VSAM 2, TSO, and POC, however, will only be furnished within the framework of Edition 7.1 of the SVS.

4. Final Remarks

In summary, it can be stated that the OC-7 EC makes available for the user of models of the ESER Series 2, especially to the user of EC 1055, EC 1055 M, or EC 1056, an operating system with new properties. Besides offering effective support of conventional batch operation, it also offers a modern, efficient, and convenient support of operating modes which are becoming more and more important, such as multimachine operation, dialogue operation, and remote processing.

For the application of problem oriented programming languages, powerful programming systems are being offered instead of the previous separate translators.

To a quite decisive extent, program throughput has been increased in the SVS operating system. As a result, the user is enabled, without changing his device technology, to process significantly more application projects in the same time as compared to using the OC-6.1 EC. When using the SVS of the OC-7 EC on a data processing system EC 1055, EC 1055 M, or EC 1056, instead of the SVS of the OC-6.1 EC, the user gains the impression that he has available a significantly faster data processing system.

This result therefore corresponds to the economic objectives that were prescribed in the Party decisions.

The OC-7 EC will become the main operating system for medium and large models of Series 2 of the ESER. It will be used for many years. Its further development will concern the expansion of its functional scope, the inclusion of new components of Type 2, especially new programming systems, the increase of user friendliness and the support of new device technology.

References:

/1/ Holenda, V.: DOS-3/ES Operating System, rechentechnik/datenverarbeitung (Computer Technology/Data Processing) 20 (1983) 12, p. 13.

/2/ Männel, K.-H. et al.: Expansion of Edition 6.1 of the OS/ES, rechentechnik/datenverarbeitung (Computer Technology/Data Processing) 18 (1981) 2, p. 20.

/3/ Lampenscherf, W., et al.: System of Virtual Machines (SVM/ES), rechentechnik/datenverarbeitung (Computer Technology/Data Processing) 18 (1981) 2, p. 23.

System of Virtual Machines, Version 3

East Berlin EDV ASPEKTE in German No 3, 1984 (signed to press 30 May 84)
pp 5-8

[Article by Dr Klaus Heinecke, Achim Schroeder, VEB ROBOTRON Center for Research and Engineering]

[Text] The SVM/ES system for virtual machines, Edition 1.2, was further developed into the SVM system for virtual machines, Edition 3.0. The following significant objectives were here achieved:

Increasing the application spectrum by including the support of more ESER models and units
Increasing the efficiency of utilization of real resources
Improving operability
Expanding the means for dialogue programming and testing
Increasing data compatability to the BPS and SVS operating systems.

Delivery will begin at the end of 1984.

1. Survey

The concept of virtual machines was explained in reference /1/, and the functional properties of the SVM/ES, Edition 1.2, were described. This system has been available since the middle of 1982 for application on the models EC 1055 and EC 1055 M of the ESER. The SVM system, Edition 3.0, is part of the operating system complex OS/ES, Edition 7, and represents a direct further development of the SVM/ES, Edition 1.2. Here, the following three essential development lines were primarily pursued:

- supporting new models and units of the ESER
- increasing the reliability, availability, and efficiency of using DP systems including the operating system and
- furnishing new functional capabilities.

The operating system SVM consists of the following components:

- control program (CP)
- dialogue-capable programming and testing system (PTS)
- trouble-shooting system (PDAS)
- remote file transmission system (RFTS)

The CP manages the resources of the real DP system and images the configuration elements of the virtual machines on the real DP system. Here, the CP guarantees that the users of the virtual machines will be independent of one another as well as guaranteeing data protection.

The PTS represents a single-user operating system, which is adapted to work on virtual machines. For this reason, it can run only on virtual machines. Under the control of the PTS, dialogue programs or other files can be efficiently constructed or changed, programs can be translated, tested, and

executed. Besides extensive functional expansions, the PTS of the SVM, Edition 3.0, contains, in addition to the features of the SVM/ES, Edition 1.2, a new procedure interpretor and a new editor with significantly increased power.

The PDAS is an extension of the PTS for the acquisition, evaluation, and management of information through error functions. The RFTS is an operating system adapted for work on virtual machines. It is used to transmit files between remotely connected batch-oriented data stations and the DP system, as well as between the data stations among one another.

SVM, Edition 3.0, supports the models of the ESER, Series 2; in particular, it can be used not only for the above-mentioned units, but also on EC 1056, EC 1035, EC 1036, EC 1045, EC 1060, and EC 1065. EC 7925 is a single-picture video device and is used for support as a virtual operating unit. The support of the external units was essentially expanded to the scope of the device technology supported within the operating system BPS and SVS. Previously, the use of the matrix module (MAMO) of the central unit EC 2655 was possible only under the control of an SVS operating system on a real DP installation. In the SVM, Edition 3.0, the support of the MAMO was effected in the CP, but here the number of simultaneously active virtual machines (VM) which use the MAMO was limited to two for reasons of efficiency. At the same time, MAMO was supported in the PTS, so that work could be performed on the development of MAMO programs in dialogue operation by means of the SVM, Edition 3.0, in parallel to using MAMO in production operations.

Finally, the support of connection processor (AP) and symmetric multiprocessor (MP) configurations of ESER central units was taken up in the CP of the SVM, Edition 3.0. The changes that have been made in the SVM, Edition 3.0, as compared to the SVM/ES, Edition 1.2, are limited to the components CP and PTS, while the components PDAS and RFTS were taken over essentially unchanged. The explanations concerning RFTS in /1/, however, should be supplemented in the direction that, by means of RFTS, it is also possible to transmit files with the means of remote data processing between several DP systems.

The most important extension of the CP and PTS of the SVM, Edition 3.0, are described in detail below; they are summarized in Figures 1 and 2.

2. Further Development of the CP

The SVM guarantees a high level of data protection. By means of segment and page tables, the CP simulates a separate virtual address space for each virtual machine. Consequently, it is not possible to address the memory of another (virtual) machine from a given virtual machine. Consequently, there is absolute read and write protection for the virtual memories of the virtual machines.

The configuration of a VM is defined in the user index and can be changed temporarily during a session only by this VM. No access to other virtual machines is permitted, unless this is expressly agreed upon. For the (always virtual) disks of a VM, there is password protection, i.e. access to

a virtual disk is possible only by entering a password. Physical disk stacks or temporary disk areas can be assigned to a VM. These temporary virtual disks exist only during one session, from their definition until their release (DETACH or LOGOFF). The content of temporary disk areas are then erased by the CP and thus are withdrawn from access by subsequent users. The CP permits only access to disk areas which are assigned to a VM.

Through the CP commands INDICATE and MONITOR, authorized users can have system events displayed, entered, and recorded. By subsequent evaluation of these data, it is possible to analyze the performance behavior. As regards the SVM, Edition 3.0, these functions were expanded, especially in view of the activity of the I/O units. To increase availability, analogous to the SVS operating system, a function for recognizing existing I/O interrupts is implemented. This secures that the CP reports to the system operator, during its cyclic investigations (every three minutes), that an I/O unit has not operated after a STOP command. In the SVM, it is possible to reserve SAVE areas on the system disk for one or more virtual machines during the generation process. If a system termination occurs or if a corresponding VM is terminated by the system operator (FORCE), the CP automatically saves the virtual memory of the SVM. Later, for example after reinitialization of the SVM, it is possible to continue the work of this saved virtual machine at the interrupt point or to analyze the virtual memory of this VM upon termination. This should be of interest for long-term jobs which cannot be concluded during the available computing time.

In the development of the SVM, Edition 3.0, various measures were implemented to increase the efficiency of this operating system. These include a means for tracing the system load and a dynamic resource allocation in dependence on this. Thus it is secured that each virtual machine on the average receives the same proportion of CPU time (ZE time). Virtual machines in which there is little I/O work receive larger time slices of CPU time at greater intervals, so that the number of alternations of allocation of the CPU to the virtual machines is reduced. The CP rolls out into an external memory tables and page management which is not addressed for a long period of time (magnetic disk), in order to relieve the working memory. The system offers the system operator and the user various capabilities to prefer one or more virtual machines with respect to and at the expense of other ones.

Figure 1: Developments in the CP

Improvements of data protection
Increase of availability
Dynamic resource management and more efficient work
Transfer of the work of one operating system on one virtual machine to work on the real system and vice versa without reinitialization
Rollout of files of the file buffer onto magnetic tape
New communication capabilities between virtual machines
Control of several virtual machines from one terminal

Thus, CPU time allocation can be influenced through the commands SET-PRIOR and SET FAVOR nn. The page change that must be performed for a virtual machine can be reduced at the cost of the other virtual machines as follows:

1. Certain pages of the virtual machine are not again rolled out.
2. A certain portion of the real memory capacity is permanently available to the virtual machine.
3. The lower real memory area is permanently assigned to a VM. It operates as a "V = R" machine. For this VM, page changes are no longer necessary.

The third possibility is especially interesting when parallel effective batch work and dialogue operation are to be implemented.

Measurements have shown that, by means of an operating system for batch work (SVS of the OC-7 EC) under certain conditions, one can achieve a throughput of about 60 percent compared to the work on a real system. This value can increase to 80 percent if the SVS system works on a V = R machine. This suggests the following regimens for efficient work on a DP system under SVM:

- batch work with SVS on the V = R machine
- dialogue work on all other virtual machines, i.e. work for program development and testing, troubleshooting, hardware troubleshooting, documentation generation and maintenance, operator training, etc.

With the command QVM, a user of a guest operating system on a VM has the capability of transferring from control under SVM to the normal operating mode. This operating system then becomes the sole user of all the resources of the DP system.

The SVM indeed continues to remain in working storage, but is no longer operative. With a restart interrupt, the system operator can cause resumption of the work of the SVM. Then the guest operating system again runs under the control of the SVM. During this change, the guest operating system need not interrupt its work.

However, the guest operating system must work in the V = R machine and, before the transfer, the work of all other virtual machines must be completed. This change of operating modes for the DP system suggests itself if efficient batch work is to be performed at night under an SVS operating system and dialogue work is to be performed by day in addition to the batch work. By means of the new SPTAPE command, it is possible to put buffered files on intermediate storage on tape. This function is of interest, for example, when the temporary space on disks, which is available to the system, is filled up so that file buffering and thus the system are blocked, or if buffered files which comprise a memory readout are not to be printed out but are to be saved for subsequent analysis.

Virtual machines operate independently of one another. Through the type of problems to be solved, however, it may become necessary to exchange information or data between virtual machines. The SVM offers the following capabilities in this connection:

- file buffering
- virtual channel-channel adapter and
- communication functions VMCF and IUCV.

The virtual channel-channel adapter permits data exchange between the main memories of virtual machines according to the same principle that applies to a real channel-channel adapter. By means of the communication functions VMCF and IUCV, direct communication is facilitated between various active virtual machines. The user imposes special requirements on VMCF and IUCV, and the exchange is programmed. By means of macros and the CP command FMSG, the user has available extensive aids for this purpose.

With VMCF, each individual transfer must be prepared and specially controlled. On the other hand, IUCV represents a higher level of information exchange. Once various paths between virtual machines have been set up, transfers in each direction can be performed quickly and in uncomplicated fashion. Through a macro IUCV, all functions can be selected, for example the setting up and cancelling of paths, or transmission of messages and responses to them.

A functional extension of the CP makes it possible to control several virtual machines from one terminal. For this purpose, virtual machines must be designated as primary and secondary users during the generation process and must be assigned to one another. If the primary user releases his virtual console (command DISCONNECT), all outputs will occur through the virtual console of the secondary user. Inversely, the secondary user, by means of the command SEND, can transmit entries to the primary user.

3. Further Development of the PTS

The PTS file system has been expanded. Thus, it is possible to use, instead of the physical record length (blocksize) of 800 bytes on PTS disks, also blocksizes of 1 K, 2 K, and 4 Kbytes. The possible number of accessible virtual disks for one virtual machine has been increased from 10 to 26. The maximum file size has also been increased from 64 Kbytes to 2^{31}-1 byte. Besides a blocksize of 800 bytes, 4 Kbytes are also permitted for a PTS file on tape.

A gap in the functional scope of the PTS consisted in the fact that only tapes without an identification record could be processed. Now, an extensive processing of ID records has been implemented, which in particular offers the following capabilities:

- the testing of standard identification records during the reading of magnetic tape files
- the writing of standard identification records
- the processing of user standard identification records
- the processing of non-standard identification records.

Tape files can be processed by means of the simulated macros of the operating systems BPS and SVS (OPEN, CLOSE, ...) or with PTS-specific means (PTS commands TAPE, TAPPDS, TAPEMAC, MOVEFILE). Before handling the identification

records, it is necessary to specify the identification record or the content of the identification record for the corresponding file by means of the commands FILEDEF and LABELDEF. As an example, we show the formation of a PTS macro library on a virtual disk (file mode A) from a file CPMAC, which is situated on a tape with the archive number DB4312, and which is unloaded and subordinate:

```
FILEDEF MACRO TAP1 SL VOLID DB4312
LABELDEF MACRO FID CPMAC
TAPEMAC CPMACRO SL MACRO
```

Figure 2. Developments in the PTS

Expanded file system
Processing of tape identification records
Conversion of PTS files into sequential and hierarchically organized files
Support of VSAM and AMS
Program push-down memory
Expanded editor XEDIT
Expanded procedure interpretor EXECE
HELP function
Processing of modules of the operating system BPS and SVS under PTS

The tape is located on the TAP1 unit (virtual address 181).

The results of processing the command sequence is the file CPMACRO MACLIB A.

Changes in sequential and hierarchical files are possible under PTS. In order to be changed, these files are brought over into PTS files. Previously, it was only possible to convert into PTS files sequentially organized files and the contents of hierarchically organized files by means of the command MOVEFILE. Now, there are two possibilities for the inverse path:

1. Use of the command MOVETSET:

By means of this PTS command, it is possible to write PTS files on a tape in the format of an unloaded file, which is suitable for return storage into a sequential or hierarchically organized file by means of the utility program IEHMOVE of the operating systems BPS or SVS.

2. Use of the command MOVEDSET:

By means of this PTS command, a sequential or hierarchically organized disk file or the content of a hierarchically organized file can be produced directly from a PTS file.

It should be noted that reblocking is the only format change which can be implemented on the target files by means of the commands MOVETSET and MOVEDSET. The access method VSAM and the VSAM utility program AMS in the PTS is supported with the objective of being able to execute and test VSAM programs under PTS.

The user has the capability of reading and setting up VSAM files. Here, data compatability with the operating systems BPS and SVS is assured. Requirements concerning the new PTS command AMSERV are transmitted through the VSAM utility program. It should be noted that, under PTS, the memory management for direct access units is not supported as in the operating systems BPS and SVS. This means that the setting up and expansion of VSAM files must take place under an operating system BPS or SVS.

There are two paths for processing loader modules from files of the operating systems BPS and SVS under PTS:

1. With the new PTS command OSRUN, one can directly call a load module from a load module library on disk, and have it processed.

An example for processing the content TEST1 of the load module library SYS1.TESTLIB:

```
FILEDEF  SYSLIB DISK OSLIB LOADLIB B DSN SYS1 TESTLIB
GLOBAL LOADLIB OSLIB
OSRUN TEST
```

2. By means of the new PTS command MOVELOAD, one can staticize object modules in a PTS disk. These object modules come from an object module library, or they have been constructed by means of the utility programs IEHMOVE or IEBCOPY of the operating systems BPS or SVS. The object modules are here converted to PTS files type TEXT and can be further processed just like these. For example, it is possible to combine several PTS files of the file type TEXT with the new PTS command PRELOAD to form one TEXT file, and thus to resolve external references. This function corresponds to the program linkage editor in the BPS and SVS operating systems. When working with PTS, a read request is present at the virtual operating unit, if execution of a command has been terminated, if a command (for example FORMAT) is waiting for a reply from the operating unit, or if a program causes a read request. The PTS offers the capability to put entry lines (commands or data lines) into intermediate storage and to use these by subsequent read requests.

Two chains of input lines are constructed here:

- program push-down storage and
- push-down storage of the operating unit.

The push-down storage of the operating unit is constructed for input lines which are entered via the operating unit, while input lines which are in intermediate storage within a program cause the construction of the program push-down storage. These input lines for the program push-down storage can be up to 512 bytes long. By means of the PTS commands MAKEBUF and DROPBUF, it becomes possible to manage several levels of input lines in the push-down storage (Figure 3).

The general read sequence in PTS is as follows:

1. Read from the program push-down storage, beginning with the lines from the last generated level.
2. Read from the push-down storage of the operating unit.
3. Physical reading of the virtual operating unit.

The editor of PTS is used to construct and change PTS disk files in the conversational mode. It makes possible the addition, finding, changing, and deletion of data records at arbitrary points in the file. By means of the PTS command EDIT, a previously existing simple editor is called. By means of EXEDIT, a newly developed expanded editor is called, which implements a significantly expanded functional scope. With the simple editor, the screen is rigidly divided and entries can be made only in a command line of the screen. But with the expanded editor, the variable utilization of the entire screen is supported. Here, a logical screen is constructed, which, among other things, contains an area for the data records, a prefix area, and a command line. Changes can be made directly in the data record. By entering subcommands in the command line or by prefix subcommands in the prefix area, a large number of functions can be implemented. The physical screen can be subdivided into several logical screens. Several files can thus be processed in parallel. By using the UPDATE function, the expanded editor records in change files the changes that are being made in a particular file.

Figure 3. Procedure for filling up the program push-down storage

```
STECK EXEC

&STACK JENE
&STACK EINGABE
MAKEBUF
&STACK ALS
&STACK LIFO EHER
&STACK LIFO KOMMT
MAKEBUF
&STACK DIESE
&STACK EINGABE
&EXIT
```

After this procedure has been processed, the read sequence of the data from the program push-down memory is as follows:

DIESE EINGABE KOMMT EHER ALS JENE EINGABE.

Previously there existed in PTS a language of procedures (EXEC procedures) with the corresponding EXEC processor. Now an additional expanded procedure language with a new processor (EXECE) is implemented in the PTS. It comprises all the capabilities of the EXEC processor without there existing syntactical compatability. EXEC and EXECE statements therefore may not be mixed, but EXEC and EXECE procedures can call one another. Significant expansions of EXECE as compared to EXEC are:

- Instead of limitation to a length of eight characters, character chains with a length up to 255 characters will be processed.
- In the EXECE processor, not only are parameters transferred from a procedure to the program called by it (procedure), but values are also transferred from the called program to the calling procedure.
- In an EXECE procedure, subprocedures can be formed, by means of which, for example, user-specific functions can be defined.
- By reducing file accesses, the processing efficiency of EXECE procedures is improved (&READ,&WRITE).
- In the EXECE processor, the most important language elements of structured programming have been included.

By means of the new HELP command of the PTS, the user can have displayed on the screen information that is necessary for his work. This includes the function, the format, the operants, and a description of the commands of the PTS and of the CP, the subcommands for EDIT and XEDIT, and the control statements of the EXEC processors EXEC and EXECE, as well as descriptions of messages from the CP and the PTS, which are not self-explanatory.

Under the control of PTS, one can translate source programs which are written in the programming languages Assembler, COBOL, FORTRAN, PL/1, and PASCAL. The corresponding programming systems can here be used by using special PTS connection modules. The programming systems are furnished on special distribution tapes, which also contain the PTS connection modules and the information auxiliary means that are required for inclusion within PTS.

References:

/1/ Lampenscherf, S.; Schröder, A.; Wagner, K.: System of Virtual Machines (SVM/ES), rechentechnik/datenverarbeitung (Computer Technology/Data Processing) 18 (1981) 2, p. 23.

BPS System for Batch Operation

East Berlin EDV ASPEKTE in German No 3, 1984 (signed to press 30 May 84) pp 9-10

[Article by Bernd Wetzel, VEB ROBOTRON Center for Research and Engineering]

[Text] Just like the SVS 7.0, the BPS was developed on the basis of the control program configuration SVS of Edition 6.1 of the OS/ES. BPS can run only on the virtual machines of the SVM. The PTS of the SVM is available to the BPS user as a subscriber system. The BPS is more powerful than the SVS of Edition 6.1 of the OS/ES on real DP systems.

The operating system BPS, just like the operating system SVS, is an upwards compatible further development of the control program configuration SVS of Edition 6.1 of the OS/ES. The main objective in the development of the BPS was to develop an operating system for batch operation, which is runnable on virtual machines of the SVM at least as efficiently as the SVS of Edition 6.1

of the OS/ES on the real DP system. BPS uses the functional capabilities of the ESER, Series 2, and especially the capabilities which the SVM guest operating system offers on virtual machines. It is not possible to use BPS independently on real systems.

The generation multiplicity of the BPS was strongly reduced compared to Edition 6.1 of the OS/ES. Functions which could previously be selected are contained as a matter of standard at the highest level. Generations in BPS are used exclusively to change the supported I/O configuration of the starter operating system. Two components of Type 2 are already contained in the starter operating system. These are the Assembler 2 as well as Sorting and Mixing 2.

Other components of Type 2 which were developed for processing under BPS control are the programming systems PL/1, COBOL, FORTRAN, and PASCAL. Some components of Edition 6.1 of the OS/ES are no longer available. These are the components Assembler, ALGOL, RPG, COBOL, and FORTRAN.

Two fulfill the main objectives specified above, the disadvantages of operating systems which were designed for working on real installations but which are operated on virtual machines were systematically analyzed. Changes were implemented which basically led to making BPS more efficient. Thus, the following important changes were undertaken:

- Support of a special virtual memory was eliminated, since the virtual memory is implemented by the control program (CP) of the SVM. Thus, the operating mode could be changed. The BC mode, which actually is typical for operating systems of Series 1, could be used by the BPS.
- The use of privileged instructions was strongly reduced and thus the corresponding simulation effort was also reduced.
- Central control program sections, such as the dispatcher, the I/O supervisor, or the program loader, were optimized.

Capabilities which are present in operating systems with virtual storage were used efficiently, and procedures which were developed for systems without virtual storage, but which work inefficiently in virtual systems, were revised. Thus, a series of measures were implemented with the objective of reducing the number of I/O operations that need to be started:

- The blocking factors for system files and files of catalogued procedures were increased. Thus, the same data quantities can be entered or read out with fewer I/O operations.
- The access methods for sequentially organized files were modified so that the most favorable number of buffers is furnished, and an I/O operation is only started after all buffer contents have been constructed or processed. In this way, the same quantities of data are also transmitted with fewer I/O operations.
- The useability of trace linkage equipment of units with direct access was expanded to loader module libraries.

- A resident index, also called resident BLDL table, was constructed for all contents of the library SYS1.LINKLIB and all libraries chained there with. This happens at system initialization time. This obviates all accesses to the indices of these libraries in the running system. Input and output operations are obviated.

Functions of the SVS of Edition 6.1 of the OS/ES, for which there exists an equivalent in SVM, were not taken over into the BPS. This applies to the functions:

- GTF (tracing aids)
- DDM (dynamic test monitor)
- CRJE (dialogue job entry) and
- TSO (subscriber operation).

In place of GTF and DDM, the CP-TRACE means are available, and, as a dialogue system to implement subscriber operation, the PTS of the SVM is available.

Also missing is the support of the file SYS1.DUMP. In PBS, the CP-DUMP command is used.

The component MMS (multimachine support) was not taken over into BPS. Work is being done on developing a multimachine support.

By means of the control program of the SVM, capabilities are offered which extend beyond the virtualization of a real machine. A few of such functions of the CP are used by the BPS:

Analogous to the PTS of the SVM, the structure of a named system is also supported in BPS. This means that the closed procedure of system initialization must be performed only once. Here, a copy of the initialized system is produced and is saved in a system that is named and furnished by SVM. With all further system starts, a very rapid initial program loading is possible from the named system.

In the standard case, all virtual machines, for which pages are lacking in the real memory, are placed in the waiting state by means of the CP of the SVM. For operating systems with their own multiprogram operations in a virtual machine, however, it is possible that further jobs exist which are in a state that can be selected for processing. By using the function pseudo-page excerpt of the SVM, the BPS is enabled to determine whether such jobs exist. Only if no selectable jobs exist, is the BPS system put into the waiting state.

The BPS uses the cabability of the SVM to exchange data between virtual machines. Here, additional virtual machines are installed in the SVM, and their address spaces are used as external data media. Magnetic tapes and disks with a capacity up to about 60 Mbytes are simulated in these address spaces. Such virtual machines are designated as pseudotapes or pseudodisks. A special type of a psuedodisk is the internal pseudodisk. With this variant, it is not the virtual memory of another virtual machine that is used, but a portion of the virtual memory of the BPS system itself.

The use of pseudotapes and pseudodisks can under some circumstances increase the efficiency of the operating system in the case of large real memories. Files on pseudounits, however, are not suited for using the components test point/restart.

The operating system BPS is currently availabile in status 7.0. Initial comparison measurements show that the main objective for its development was overfulfilled. Using a measurement job stream on the EC 1055 M, a throughput increase of 45 percent was achieved under specified comparable conditions and when using psuedodisks and internal pseudodisks. When using real data media, the throughput was increased up to 30 percent compared to the SVS of Edition 6.1 of the OS/ES on a real DP system.

Application of the BPS does not have high priority in the GDR.

SVS 7.0 Operating System

East Berlin EDV ASPEKTE in German No 3, 1984 (signed to press 30 May 1984) pp 10-19

[Article by Dr Karl-Heinz Maennel, Siegfried Schneider, VEB ROBOTRON Center for Research and Engineering]

[Text] This article circumscribes the functional scope of the operating system SVS 7.0 as a component of the operating system complex OC-7.0 EC and explains the reasons and objectives of the development.

Starting from the SVS of the OC-6.1 EC, the essential further developments of the SVS 7.0 are presented, and the achieved efficiency improvements are elucidated in terms of detailed measurements. The job profile, the measurement means, and the measurement methodology, which are used to evaluate the efficiency of the SVS 7.0, are elucidated. For problems of batch processing, a throughput increase of about 100 percent was measured under the described conditions, in comparison to the SVS of the OC-6.1 EC, modification 8. The rapid and broad introduction of the SVS 7.0 into practical application thus has extraordinarily high economic significance.

1. Objectives of the Development of the Operating System SVS 7.0 in OC-7 EC

The SVS 7.0 of the OC-7 EC represents an upward-compatible further development of the control program configuration SVS of the OC-6.1 EC. For the SVS 7.0, a self-sufficient control program was developed while simultaneously implementing new and modern designs. The modular structure of the SVS control program, including the structure of the loader module libraries, was revised as compared to the OC-6.1 EC. The SVS 7.0 control program has been comprehensively developed compared to the control program configuration SVS of the OC-6.1 EC, and has been supplemented by important novel developments for practical application. The main objectives of developing the SVS 7.0 was to improve clearly the efficiency of this operating system compared to the OC-6.1 EC, while using the functional capabilities of the models of the ESER, Series 2. To increase efficiency, newly developed functions and components

were included in SVS 7.0, which simultaneously secure a high degree of compatability with previous OS/ES applications. In the SVS 7.0, as compared to the OC-6.1 ES, the large generation multiplicity was restricted. By means of the SVS 7.0, the independently deliverable components of Type 2 were simultaneously further developed. These further developments on the one hand are used to increase efficiency and on the other hand, the functional scope of these components was also expanded. Sorting/Mixing 2 and Assembler 2 were here emphasized. The further development of these components, however, was also necessary for adaptation to the control program of the SVS 7.0.

The SVS 7.0 is completed by the following components which can be distributed independently:

Assembler 2, Edition 2.1; Sorting/Mixing 2, Edition 1.4; PL/1 programming system, Edition 2.2; COBOL programming system, Edition 1.2; FORTRAN programming system, Edition 1.1; PASCAL programming system, Edition 2.0. For the SVS 7.0, The SVM 3.0 is available to implement subscriber operation. The SVS 7.0 can run in the V = R machine of the SVM 3.0 on the models EC 1055 M and EC 1056, with only a slight reduction of throughput. Here it is presupposed that adequately large real memories are available. With the SVS 7.1, TSO SVS will be available for subscriber operation, and thus an upwards-compatible use will become possible for TSO outputs which occur in the framework of the OC-6.1 EC.

Furthermore, with the SVS 7.1, the component POC is made available to support batch processing on multicomputer complexes of the models EC 1055, EC 1055 M and EC 1056. The POC for SVS 7.0 is upwards-compatible with the POC of the OC-6.1 EC.

In the SVS 7.0, TCAM is for the first time being furnished at the level TCAM/NF (TCAM with network function) as an access method for remote data processing.

For SVS 7.1, the increase of the level of VSAM is furthermore accomplished by the development of VSAM2. In SVS 7.0, VSAM at first will still be available at the present level.

Because of the available more efficient and functionally more comprehensive components of Type 2, the following components of the OC-6.1 EC will no longer be furnished in the SVS 7.0: Assemblers and compilers of the programming languages PL/1, COBOL and FORTRAN.

In the SVS 7.0, the ALGOL compiler and the components DDM and CRJE also will no longer be furnished.

Within the operating system complex OC-7 EC, the SVS represents a self-sufficient independently marketable operating system. This means it has its own distribution system and its own generation means. The operating system SVS is a component of operating system developments coordinated within ESER and is being developed by stages with an increasing functional scope in joint collaboration with the USSR and is being furnished for application.

The SVS 7.0 will be available at the beginning of 1985 for users of the models EC 1055, EC 1055 M, and EC 1056. It is also intended for use on Soviet models of the ESER, Series 2. Support of external units is implemented in the SVS of the OC-7 EC first of all at the level of the model capacities of models EC 1055, EC 1055 M, and EC 1056, as well as the supported Soviet models.

2. The Efficiency of the OC-6.1 EC

The OC-6.1 EC was developed with the objective of supporting the models of Series 1 and Series 2 of the ESER by a common operating system. In addition to the control program configurations MFT and MVT, which were already present in the OC-4.1 EC, the control program configuration SVS was included in the control program of the OC-6.1 EC to support the models of Series 2. The transition from models of Series 1 to models of Series 2 of the ESER is thus greatly facilitated, because the control program configurations MFT and MVT for the models of Series 2 of the ESER have been and continue to be offered for use. Many of the expanded functional capabilities of the models of Series 2 can be used only in the SVS.

The increased control effort to implement new functions (e.g. the virtual memory) in the SVS could be compensated by divising procedures for the compilers, for the assembler, and for system input and output programs - as will be explained in more detail.

The comprehensive utilization of the advantages of virtual memory in the SVS of the OC-6.1 EC was only incompletely achieved on the user program level, however. Users with a high requirement for CPU time observed throughput reductions with SVS as compared with MFT and MVT.

Problems of throughput reduction were analyzed. The results of this analysis are summarized in this point.

2.1 Utilization of the main resources in the OC-6.1 EC

For operating a DP system of the ESER, the following main resources are needed in principle:

- central unit;
  by this are understood more generally the processing functions of the central unit (instruction processing)
- memory;
  by this is understood the working storage of the electronic data processing system with respect to MFT and MVT as well as the virtual memory and the real memory in the SVS
- I/O system;
  by this is understood the totality of technical resources required to implement an I/O operation, i.e. channels, control units, and external units.

These main resources are managed and controlled by the particular operating system.

The models EC 1055, EC 1055 M, and EC 1056 are designed so that the rate of data transmission is essentially determined by the speed of the external units. When analyzing the OC-6.1 EC, the workload of the main resource, namely the central unit, was found to be determinative of throughput. This main resource will be considered in more detail below.

2.2 The Main Resource: The Central Unit

The use of the central unit can be characterized by the consumed central unit time called CPU time for short. The CPU time used during the total time T can be divided into the following components:

- CPU time in the problem program status with protection code $\neq 0$, PPN time or PPN for short.
  This CPU time is required to process the instructions contained in the user program and the instructions included in the problem program in virtue of the operating system.
- CPU time in the supervisor status, called for short SV time or SV.
  This CPU time is required for supervisory services which are initiated by user programs, service programs, and the control program of the operating system. The control program times which are needed for handling I/O interrupts, external interrupts, and program interrupts likewise are included within this CPU time.
  To implement an I/O operation, for example, the supervisor calls SVCO (EXCP) and SVC1 (WAIT) are given in the problem program and initiate corresponding services in the supervisor and I/O supervisor.
  The interrupts connected with I/O operations are likewise handled in the supervisor and I/O supervisor.
- CPU time in the problem program status with protection code = 0, for short CPO time or CPO.
  This CPU time is required for processing programs involving job management. The job management services are initiated through job control statements and through commands.

According to the above, one has the following relationship:

$$CPU = SV + PPO + PPN$$

The total time T is composed of the CPU time and waiting time. During the waiting time, the central unit is in a wait state. The waiting time is composed of the waiting time that is superposed on I/O work, and the waiting time which arises from non-use of the entire system.

Implementation of the PPN times entails implementation of PPO and SV times due to the activated job management services and supervisory services. The magnitude of the PPN times is determined by the user program or the service program itself, by the access methods employed, and by the compilers. It appears that the PPN times in the OC-6.1 EC are equally great regardless of the control program configurations that are actually used. The degree of multiprogram operation likewise does not influence the PPN times.

However, another behavior is observed as regards the SV and PPO times. It appears that these times depend very strongly on the particular control program configuration in the OC-6.1 EC and on the job management and supervisory services required by the user programs. The most significant difference can be observed between SVS on the one hand and MFT and MVT on the other hand. The reasons for this are primarily the introduction of the virtual memory and the associated time-intensive running sequences in the control program of the operating system.

As an example, the average times required for important SVC of Type 1 will be cited (Table 1). Here, the extension factor for the SV time to implement an I/O operation should be especially noted. The extension is caused by the necessity of fixing the I/O areas, the control blocks, and the channel program conversion before the I/O operation.

Table 1. Comparison of the required SV times, between MVT and SVS of the OC-6.1 EC, for important supervisory calls. The time specifications represent averages.

	MVT OC-6.1 EC	SVS OC-6.1 EC	Extension factor
EXCP SVCO	1,2 ms	5,9 ms	4,9
SVC4	0,8 ms	1,9 ms	2,4
GETMAIN SVC10	0,6 ms	1,6 ms	2,7

By reducing the required supervisory services in the service programs and by suitably changing the control programs of the SVS 7.0, the SV time could be strongly reduced (see Section 4).

As the example shows, more supervisor time is expended for important supervisor calls of Type 1. This also happens when transferring from MVT to SVS, in other frequently used segments of the control program.

When transferring from MVT to SVS, the supervisor time is approximately doubled while the average user profile of batch processing remains about the same. The same circumstances can be observed when using TSO in the SVS as compared to using TSO in the MVT. Here, too, the supervisor time expended in the SVS is approximately doubled for the same tasks.

On the other hand, the use of virtual memory offers good possibilities for reducing supervisory services however. In particular, this can be achieved, in connection with using SVS, by increasing the blocking factors of data on tape and disk. Thus, the supervisory services for the starting of I/O operations, for I/O interrupt handling, and for job selection can be strongly reduced. Table 2 shows teh influence of the blocking factors on the required SV time for the SVS of the OC-6.1 EC. From this it can be seen that, when using the SVS, one should use block sizes equal or greater than 4 Kbytes. For the SVS of the OC-6.1 EC, changes were therefore implemented on the codelivered procedures for compilers, sorting/mixing, and for the assembler, in order to increase the block sizes.

Table 2. Influence of blocking factors in the SVS of the OC-6.1 EC, 8th Mod.

(Output of 30,000 records of length 200 bytes and 200 Mbyte replaceable disk storage, using the access method QSAM and the utility program IEBDG)

blocksize (byte)	200	400	1000	2000	4000	6000
number of blocks	30 000	15 000	6000	3000	1500	1000
total running time (%)	100	53,6	23,8	13,5	8,5	7
CPU time (%)	100	59,8	30,3	20,1	15,5	13,4
Waiting time (%)	100	47,6	17,6	7,1	1,8	0,9
SV time (%)	100	55,9	23,6	12,5	7,6	5,2
PPN time (%)	100	89,2	81,1	78,4	75,7	75,7
I/O interrupts (%)	100	50,3	20,4	10,4	5,4	3,8

Table 3 shows the results of a comparative measurement between MVT and SVS of the OC-6.1 EC. This comparison measurement refers to an average batch processing profile.

The system as generated for this measurement contained the compilers, sorting/mixing, and assembler proper to the OC-6.1 EC, and not the corresponding components of Type 2. For this reason, the job stream described in Section 4 also could not be used for this comparison.

The main portion of the increase in supervisory time is based on increased effort involved with the implementation of I/O operations. The entire SV time expenditure for implementing I/O operations has approximately doubled in the SVS of the OC-6.1 EC. The waiting time associated with MVT utilization can normally be used for the increased expenditure of supervisory time in the SVS. However, if small portions of waiting time are already present in the total running time when using MVT, the total running time in the SVS is necessarily increased, and thus the program throughput is reduced. For comparing the work of two operating systems on the same hardware, the program throughput achieved with the operating systems is an important parameter.

By program throughput, one understands the processed amount of program per unit time. This amount of program depends on the power of the operating system and on the application profile, if one presupposes the same hardware resources. For batch processing, the number of process jobs in a certain time is equivalent to the program throughput.

To compare the program throughput of two operating systems, the following method was used:

In one operating system, a job stream with a previously fixed application profile was processed and the run through time T_1 required for this was

Table 3. Comparison of the efficiency of the control program configurations MVT and SVS of the OC-6.1 EC when using 29 Mbyte and 100 Mbyte WPS

29 M Byte	MVT		SVS mit Standardprozeduren [1]		SVS mit SVS-spezifischen Prozeduren [2]	
Gesamtlaufzeit [3]	1814 s	100 %	1774 s	98 %	1335 s	74 %
SV-Zeit [4]	602 s 33,2 %	100 %	1091 s 61,5 %	181 %	786 s 58,9 %	131 %
PPN-Zeit [4]	309 s 17 %	100 %	321 s 18,1 %	104 %	307 s 23 %	99 %
W-Zeit [4]	874 s 48,2 %	100 %	333 s 18,8 %	38 %	213 s 16 %	24 %
PPO-Zeit [4]	29 s 1,6 %	100 %	29 s 1,6 %	100 %	29 s 2,2 %	100 %
EXCP + EXCPVR	44585	100 %	43183	97 %	24364	55 %
100 M Byte						
Gesamtlaufzeit [3]	755 s	100 %	705 s	93,4 %		
SV-Zeit [4]	228,8 s 30,3 %	100 %	449,8 s 63,8 %	196,6 %		
PPN-Zeit [4]	160 s 21,2 %	100 %	172,7 s 24,5 %	107,5 %		
W-Zeit [4]	348,8 s 46,2 %	100 %	63,4 s 9 %	18,2 %		
PPO-Zeit [4]	17,4 s 2,3 %	100 %	17,6 s 2,5 %	101 %		
EXCP + EXCPVR	26378		24350	92,3 %		

1 SVS with standard procedures
2 SVS with SVS-specific procedures
3 total running time
4 time

measured. Under identical hardware conditions, the same job stream was processed under the control of the other operating system, and the run through time T_2 was measured. If T_1 is greater than T_2, the increase of

program throughput is derived from the program quantity of the same application profile which can additionally be processed in the time $T_1 - T_2$, which means $(T_1 - T_2)/T_1$ times the processed program quantity. The increase of throughput is therefore $(T_1/T_2) - 1$.

If the waiting time remains the same and if the PPN of the two operating systems remains constant, one obtains the following relationship for the increase in throughput:

$$D = \frac{C}{1 - C} \quad \text{for } T_1 > T_2 \text{ and} \qquad C = \frac{(SV_1 - SV_2) + (PPO_1 - PPO_2)}{T_1},$$

where SV_1 and SV_2 are the SV times and PPO_1 and PPO_2 are the PPO times for the respective operating systems.

The measurements were made without evaluating system input and system output.

For measuring the efficiency of 100 Mbyte WPS, the system was left at the 29 Mbyte WPS. All other files were located on 100 Mbyte WPS.

It should be noted that a significant portion of I/O operations is generated through the control programs of the operating system itself, in implementation of functions that are explicitly or implicitly required by the usef. Upon transfer to the SVS in the OC-6.1 EC, and when using the virtual memory, I/O operations for program loading were reduced. Thus, bottlenecks which exist in the MFT and MVT and which lead to the waiting times customary there were simultaneously eliminated in the SVS.

The utilization of the capabilities of the virtual memory by the control programs of the SVS in the OC-6.1 EC is still only incomplete, however. Thus for example, the modular structure of the control program remains essentially intact when transferring from the MVT to the SVS of the OC-6.1 EC. At the same time, the dynamic program calls via the supervisor macros XCTL, LINK, and LOAD thus remain preserved, although, with full utilization of the virtual memory, the restrictions of the memory sizes useable for control program sections could be essentially eliminated. Similar circumstances prevail with the memory requirements in the SVS of the OC-6.1 EC via GETMAIN and FREEMAIN. Compared to MVT, the control programs are essentially unchanged as far as storage requirements are concerned, although the above-mentioned memory restrictions for control programs can be essentially eliminated.

3. Further Development of Control Programs in the SVS 7.0

3.1 General Objectives

Analysis of the system behavior of the SVS in the OC 6.1 EC showed that the most important objective for the development of the SVS 7.0 was to reduce the required expenditure of supervisory time while simultaneously reducing the workload on channels, control units, and devices. A reduction of supervisory time then entails an increase of throughput for an average batch processing profile.

To reduce the supervisory time, the SVS 7.0 implemented new principles of control in the operating system under comprehensive and maximally optimized utilization of the operating principles of ESER, Series 2. The following main methods were used:

- Reduction of the number of required supervisory services in the control program, in the utility programs, and in the compilers
- Replacing laborious supervisory service by more efficient ones
- Developing new and more efficient functions and components
- Optimization of frequently occurring control program sections.

After the development of the SVS 7.0 was completed, an evaluation was performed by comparison measurements, based on an average application profile of batch processing. We shall make further reference to the results of this comparison measurement.

The SVS 7.0 furthermore furnished means for measuring system workload. This enables the user to control system workloads.

Functionally, the control program of the SVS 7.0 was supplemented as compared to the OC-6.1 EC by new and expanded functions in job management, an integrated job network control and terminal time control as well as by supporting the timer potentialities of the models of ESER Series 2.

3.2 Reduction of the number of required supervisory services in the SVS 7.0

The load module structure of the control program of the SVS 7.0 was modified in such a fashion that the modules are collected together to the largest possible load module (linkage), and the XCTLs were replaced by direct branchings. By evaluating the above-mentioned comparison measurements, it can be observed that the number of required XCTLs has been reduced to a total of about 3 percent.

By changing the memory requirements in the control programs, especially in the programs of job management and data management, the number of GETMAINs and FREEMAINs could be reduced to about 23 percent. In the control program sections of the SVS 7.0, which operates in the supervisor status, the principle of requiring supervisory services by directly branching to the corresponding supervisory chains was increasingly used. In this way, the number of EXITs with reference to the above-described comparison measurement could be reduced to about 26 percent.

The increase of average block size for files on tape and disk likewise caused a reduction of the number of supervisory services (EXCP, EXCPVR, WAIT).

By increasing the memory area used for the compiler, Sorting/Mixing 2, and Assembler 2, it is possible to maintain largely in virtual memory the intermediate data which previously had to be stored in temporary files. This is achieved by increasing the size of regions in the procedures and by special modifications of the Assembler 2, Edition 2.0. In this way, the number of required supervisory services could likewise be reduced. The number

of SVC interrupts in the SVS 7.0, as compared to the OC-6.1 EC, could be reduced down to about 40 percent.

3.3 Replacement of laborious supervisory services by more efficient ones

A special increase of supervisory time was achieved by supporting the virtual memory in connection with supervisory services for the implementation of I/O operations. This is based in the control program functions that must be additionally executed. Essentially, these are the following functions:

- Construction of address lists of the pages that must be fixed to execute the I/O operation, by means of the I/O supervisor for all the required control blocks, for the channel program, for the required follow-on routines, and for the data areas. For the channel program, a real copy is constructed through channel program conversion.
- The fixing and unfixing of pages is effected before and after each I/O operation by means of the page supervisor - including the loading of pages should this be required.

When the frequency of I/O operations is appropriately high, the fixed and then unfixed pages remain preserved in the real memory and most likely will not be rolled out. It therefore lies close at hand to place the laborious fixing and unfixing processes as well as the channel program conversion as much as possible into the OPEN and CLOSE phase, and thus to execute them only once for one file.

If it can be guaranteed that the required memory areas for the implementation of an I/O operation are fixed, and that the channel program conversion has taken place, one can use EXCPVR instead of EXCP in the control programs. As a result, the above-mentioned additional functions in the I/O supervisor and page supervisor are not executed. The required pages can be fixed at the latest in the SIO follow-on routines of the access methods, whereby it becomes possible, for example, possible to take into account data regions which may be subject to change.

In the SVS 7.0, EXCPVR was introduced for the following programs:

- QSAM and BSAM
- accesses to the job chain file in job management
- SPAM
- Sorting/Mixing

For the access methods QSAM and BSAM, one obtains the following savings in supervisory time:

	QSAM	BSAM
EXCP (SVS OC-6.1 EC)	5 ms	5 ms
EXCPVR (SVS 7.0)	1.9 ms	2.9 ms
Remaining time expenditure	38 %	58%

The higher expenditure for BSAM arises from the fact that the data areas cannot be fixed in the OPEN phase. They are fixed in the supervisory follow-on routine. If the positions of the data areas remain the same, the same savings in supervisory time can be achieved for BSAM as for QSAM.

In the measurement job stream, the portion of accesses to sequential files is about 75 percent. This corresponds to the internationally determined proportion of these access methods in actual application. In the measurement job stream, the BSAM accesses are about twice as high as the QSAM accesses.

In the SVS 7.0, the job chain accesses occur mainly through track buffering. When analyzing the supervisory time required for track buffering, the channel program conversion proved to be very laborious. By introducing EXCPVR, the supervisory time required for the programs of job management could be considerably reduced.

In the development of the new access method SPAM, accesses to the SPOOL file as well as accesses to the job chain file are implemented via EXCPVR.

In evaluating the measurements, making a comparison between the SVS of the OC-6.1 EC and the SVS 7.0, the following change of the proportion of EXCP and EXCPVR as well as of the required time could be observed:

	Number		Required time
	EXCP	EXCPVR	EXCP + EXCPVR
SVS des OC-6.1 EC	98 %	2 %	100 %
SVS 7.0	32 %	68 %	20,6 %

The number of required page changes is changed only insignificantly during the transition from the SVS of the OC-6.1 EC to the SVS 7.0, when the measurements were evaluated.

In Sorting/Mixing 2, Edition 1.4, the I/O operations for the working files (SORTWK) were likewise converted to using EXCPVR.

3.4 Revision of frequently used or time-intensive parts of the control program

The SVS of the OC-6.1 EC should also be runnable on small models of the ESER with 512 Kbytes real memory. Consequently, a decision was made in favor of a page size of 2 Kbytes. When developing the OC-7 EC, a uniform page size of 4 Kbytes for all parts of the operating system complex was introduced. This simultaneously caused a reduction of the supervisory time as a result of the shorter page tables and it also reduced the page changes.

The models of Series 2 of ESER can work both in the BC mode and in the EC mode. (In the BC mode, the system of Series 2 works like a system of Series 1. Only when using the EC mode can all the new functions of Series 2 be utilized.) Because large portions of the MVT supervisor were taken over in the development of the SVS supervisor in the OC-6.1 EC, the latter is able only to analyze PSWs in the BC mode. But to be able to use the virtual memory and other new

functions, it was not able to dispense with the work of the programs in the EC mode. For this reason, with every interrupt, the old PSW must be converted from the EC into the BC mode. When leaving the supervisor, a conversion in the inverse direction was necessary. With the new development of the supervisor SVS 7.0, these time intensive conversions are obviated.

The TRACE function in the supervisor is a necessary device so that the necessary documentation can be produced when errors occur. In the OC-6.1 EC, the TRACE function could be activated or deactivated only during system start. In practice this means that the TRACE function must always be active. In the SVS 7.0, the activation and deactivation of the TRACE function is dynamically accomplished by a command that can be given at an arbitrary time. Thus the user can work without TRACE in normal applications, which leads to a considerable gain of efficiency.

In the OC-6.1 EC, three different control program configurations could be selected from one distribution system: MFT, MVT, and SVS. This was necessary since this operating system had to support the models of Series 1 and Series 2 of the ESER. The new operating system complex OC-7 EC is designed only to support models of Series 2. For this reason, all MFT- and MVT-specific control program sections could be removed from the SVS 7.0. This not only caused a considerable speed up of the system generation of the SVS 7.0 as compared to the SVS of the OC-6.1 EC, but also reduced the size of many modules.

3.5 New components and functions of the SVS 7.0 to increase the efficiency of the overall system

3.5.1 Access method SPAM

The new access method SPAM was made available in the SVS 7.0. The access method SPAM was specially developed for temporary files and, on the user level, is compatible with BSAM and QSAM (e.g. when writing files in job control statements and procedures or in macros for the reading and writing of records or blocks). The advantage compared to BSAM and QSAM files consists in the fact that files generated by SPAM, the so-called SPOOL files, automatically are blocked high and thus require less space on the direct access memory. The access method SPAM requires a very low amount of control programming. The advantages of SPAM are especially conspicuous in comparison to the access method BSAM.

An important application of the new access method is the intermediate storage of the SYSIN and SYSOUT data in SPOOL files of the SVS 7.0.

During system start, the user can decide whether temporary files are to be set up in the form of SPOOL files. If SPOOL files are used, a SPOOL area with 4 Kbyte large blocks is formatted on one or more direct access data media. The necessary size of the SPOOL areas is directed in accord with the needs of the user.

With the access method SPAM, the data are always collected together into blocks 4 Kbytes in size. When outputting into the SPOOL area, three 4 Kbyte blocks

are always chained and are written by means of track linkage and by using EXCPVR.

The writing of data into a formatted file entails significant advantages as regards processing speed and relieves the device control unit as well as the channels. During writing, the 4 Kbyte blocks are arranged in such a way that the least number of access arm motions is required.

A SPOOL file is a data set which is written between the opening (OPEN) and the closing (CLOSE) into the SPOOL area which means that a SPOOL file consists of one or more 4 Kbyte blocks.

As a matter of standard, three 4 Kbyte large buffers are made available for one SPOOL file. The read-in size is correspondingly enlarged by the control program. The number of buffers can be changed during system start. If a record is still in the buffer during read, because of a previous read or write operation, this record is made available directly from the buffer. This mode of operation is especially favorable for file updating.

Special attention was paid to data protection. All information concerning the SPOOL is recorded in a session file which is constantly updated. This makes possible an essentially loss-free restart of the system at an arbitrary time.

To utilize the advantages of the new access method beyond the framework of the SYSIN and SYSOUT data, the procedures were revised so that temporary files are set up as much as possible as SPOOL files.

Table 4 compares the results between the access method QSAM in the OC-6.1 EC and SPAM in the SVS 7.0. Thirty thousand records of 200 byte length with various block sizes were here written into a disk file. Especially with small block sizes, the considerably lesser need for supervisory time becomes apparent.

System input and system output programs of the SVS 7.0 can process both sequential and SPOOL files. Besides the system output program, an output processor was developed especially for outputting SPOOL files. This processor works with chained channel commands. By chaining the channel commands, the requirements on the I/O supervisor are significantly reduced. The output processor also aids the floppy disk unit EC 5075.

The considerably reduced effort for intermediate storage with spool files in the SVS 7.0 as compard to sequential files in the OC-6.1 EC is expressed in the demand for CPU time. In Table 5, the CPU time required for intermediate storage of comparable data quantities is compared for QSAM and SPAM. Here it should be noted that the system input program (when reading from tape) uses up only 36 percent of the CPU time and the output processor uses up only 18 percent of the CPU time that was necessary in the OC-6.1 EC.

Table 4. Comparison of the access methods QSAM OC-6.1 EC and SPAM SVS 7.0 (Output on 200 Mbytes WPS)

(The OC-6.1 EC times always correspond to 100 percent)

Blockgröße* [Byte] 1	200	400	1000	2000	4000	6000
Blockanzahl* 2	30 000	15 000	6000	3000	1500	1000
ZE-Zeit [%] 3	13,8	22,2	41,5	61,4	78	90,3
SV-Zeit 4 [%]	1,8	3,9	8,2	15,6	25,5	34,7
PPN-Zeit 4 [%]	95	97	100	100	100	100
Anzahl der E/A- 5 Unterbrechungen [%]	2,1	4,1	10	19,5	37,1	53,2

*The blocksize and the number of blocks correspond to the values for QSAM of the OC-6.1 EC.

1 blocksize
2 number of blocks
3 CPU time
4 time
5 number of I/O interrupts

Table 5. Comparison of the system input and system output of the OC-6.1 EC, SVS, and SVS 7.0
(The OC-6.1 EC times and the number of I/O operations in the OC-6.1 EC respectively correspond to 100 percent)

(%)	SVS 7.0 system input	SVS 7.0 system output
SV time	28.4	10.3
PPO time	113	112.5
CPU time	35.9	17.5
I/O operations, number	67.6	8.5
Proportion of EXCPVR in SVS 7.0	85.4	55.5

Table 6. Comparison of running time of system input and system output

	System input			System output	
	OC-6.1 EC RDR	OC-6.1 EC RDR3200	SVS 7.0 RDR	OC-6.1 EC WTR	SVS 7.0 OPR
running time (s)	880	510	380	980	870
SV time (s)	177	152	15	188	20
PPO time (s)	17	15	14	12	19

Table 6 compares the times for reading in about 6000 punched cards and for printing about 17000 lines in the OC-6.1 EC and the SVS 7.0.

When using SPOOL files for intermediate storage, protection and service are also increased. For example, in connection with system output, all job information belonging to a job and SYSOUT files can be outputted multiply, several copies of a SYSOUT file can be produced and running printer outputs can be advanced or reset by a certain number of printed pages.

3.5.2 Standard chaining of channel programs when using the access method QSAM

An important method for reducing the number of I/O operations consists in chaining the input or output of data. If this chaining is not performed dynamically by the I/O supervisor as a function of I/O operation in the queue which are waiting to be started, but are performed as a matter of standard by the access method, only one EXCP or EXCPVR and only one WAIT is always necessary. Starting and terminating of I/O operations is effected only for a channel program. As a result, the required SV times are reduced in inverse proportion to the chaining factor. In the SVS 7.0, this type of chaining was introduced for the access method QSAM for tape and disk, for the record formats F, FB, and V. A maximum of 15 data blocks can be inputted or outputted in chained fashion. The physical block length here corresponds to the specified block length. As a result, it is possible to process small-block sequential files with high efficiency. The required I/O areas in memory are made available in 4-Kbyte blocks during the execution of OPEN. A maximum of three 4-Kbyte blocks can be made available for one file. Thus, even with a block length of 4-Kbytes, a chaining of three data blocks occurs for a 100 Mbyte WPS; for an 800 byte block size, a chaining of 15 data blocks will occur.

Table 7. Comparison of the access method QSAM in the OC-6.1 EC (SVS) and SVS 7.0 with standard chaining

(Output of 30,000 records of length 200 bytes on 200 Mbyte WPS using the utility program IEBDG; the OC-6.1 EC times always correspond to 100 percent; likewise the number of I/O interrupts in the OC-6.1 EC corresponds to 100 percent)

1 block size
2 number of blocks
3 CPU time
4 time
5 number of interrupts

Table 7

Blockgröße 1 (Byte)	200	400	1000	2000	4000	6000
Blockanzahl 2	30 000	15 000	6000	3000	1500	1000
ZE-Zeit [%] 3	18	23,2	37,2	52,7	66,5	79,3
SV-Zeit [%]	6,8	7	8,4	13,3	20	36
4 PPN-Zeit [%]	102	100	100	100	100	100
4 W-Zeit [%]	13,1	7,6	19,5	41,7	–	50
Anzahl der 5 Unterbrechungen [%]	8	9	10	18,4	35,1	94

Table 7 compares QSAM of OC-6.1 EC and the SVS 7.0 with respect to efficiency.

During inputs, the chaining factor that has been set during OPEN is almost completely effective; on the other hand, during output, the channel program is interrupted at the end of the track and is again set up for the next track with a specified chaining factor. For outputting data with the access method QSAM in the SVS 7.0, it should be noted that I/O operation is only started when the data blocks provided for the chained output are ready.

In the SVS 7.0, sequential files should preferably be processed with QSAM because of increased efficiency.

Because of the chained input and output of data, there exists a significant reduction of occupation times for the channel, the device control unit, and the device itself. Even with a 100 Mbyte WPS, the channel occupation time with 10-fold chaining is still reduced to about half. The device occupation times are reduced even more strongly. Here, the chaining factors become fully effective, which means that the device occupation time is reduced to about one tenth with 10-fold chaining. For a 29 Mbyte WPS, the reductions of the channel occupation times are still much higher.

3.5.3 Making modular loading more effective

For searching through the SYS1.LINKLIB index, when loading a module from this library, rather long occupation times of the control unit occur in the OC-6.1 EC. In the SVS 7.0, the index entries of the entire SYS1.LINKLIB chain can optionally be set up in the virtual memory. This is done during system initialization. When loading a module, the index is searched according to a hash method, in order to find the appropriate index entries.

If the module being loaded is not found in an index which is situated in virtual memory, the indexes of the SYS1.LINKLIB chain are searched. In this way, it is made certain that, even while the system is working, modules taken up into the SYS1.LINKLIB can be loaded.

3.5.4 Introduction of the SYS1.LPALIB

Setting up the system area for resident loading modules (LPA) is one of the most time consuming operations when starting a system with virtual memory. All modules which are brought into this area are situated in the library SYS1.SVCLIB, with the SVS of the OC-6.1 EC. Since not all modules of the SYS1.SVCLIB may be brought into the system area for resident loading modules, additional tables must be carried and interrogated. Thus the renewed acceptance of LPA modules was made more difficult.

Therefore, in the SVS 7.0, a new library was set up for the LPA modules, the SYS1.LPALIB. All modules contained in this library are loaded into the system area for resident loading modules during system start (cold start). In this way, the system start and the renewed acceptance of LPA modules can be speeded up. Furthermore, the SYS1.SVCLIB was significantly reduced by this measure.

Since the SYS1.LPALIB is required only during a cold start of the system, it can be set up on a separate data medium, which is not needed during a warm start of the system.

3.6 Expanding the Control Program Function in the SVS 7.0

3.6.1 System load measurements

The function of system load measurement (SLM) was included in the control program of the SVS 7.0 in order to measure and evaluate performance. This function can be activated or deactivated for the operating system by means of an operating command. It is used for the global measurement of resource loading and thus the overall system load for job profile. The measured data are entered into a file which can be evaluated by means of an analysis program existing in the operating system. User-specific analysis programs are also possible. The acquisition of measured data can be controlled as regards its scope or the type of acquisition, in terms of intervals and/or sums. The function of system load measurement includes separately its own need for resources.

On the basis of the load generated by this function itself, it is recommended that it be activated only for purposes of performance measurement and evaluation. Without including the I/O peripherals, the extra demand is at most 5 percent; if the I/O units are included, the figure is 7 percent.

The following resources can be dealt with:

1. CPU time, coded according to SV time, PPN time, PPO time, and waiting time. The waiting time is recorded separately according to waiting times superposed with I/O work, and waiting time without superposed I/O work (useless waiting time).

2. Occupation times of the devices, coded according to individual devices. Number of SIOs for the individual devices.

3. Occupation times of the channels, including the length of the queues at the logical channels, as well as causes for hindrances (occupied units, control units, or channels).

4. Number of interrupts, organized according to interrupt categories.

5. Degree of workload of the memory and workload of the virtual memory (page changes, subdivided according to reading and writing of pages, reclaimed pages).

6. Number of SIOs including condition codes.

7. Number of executed SVCs, separated by the SVC numbers.

The corresponding measured values can be determined in specified intervals and/or in summary for the entire measurement period. By evaluating the results of the performance measurements, the user can take deliberate measures to increase throughput (tuning).

3.6.2 Support of the timer equipment of ESER models of Series 2 and securing the correct job invoicing

In the SVS of the OC-6.1 EC, an interval timer is used for measuring time intervals; the CPU timer and clock comparator are not used by the timer routines of the control program.

In the SVS 7.0, the clock makes available the data and the time of day. The clock comparator and CPU timer are used to measure and perform time intervals, which achieve a significantly higher accuracy in the measurement and formation of time intervals.

In the OC-6.1 EC, the CPU time required for an I/O interrupt or a page exception is accounted to the job which was active just at the time of the interrupt. The relatively unsharp time measurement over the interval timer as well as the allocation of the CPU time for treating I/O interrupts had as a consequence that the CPU time assigned by the control program to a job step fluctuated severely.

In the OC-7 EC, only the CPU expenditure for SVC interrupts is assigned to the active job.

4. Comparison of the Throughput of the OC-6.1 EC, SVS, with SVS 7.0

The throughput for batch processing which can be achieved with the SVS 7.0 was determined by measurement according to the method specified under Point 2.

The measurements refer to comparison runs of a job stream which was processed under the control of the SVS of the OC-6.1 EC and the SVS 7.0. These measurements represent averages. The job stream used for the comparison run was selected on the basis of questionnaires in order to obtain the most representative possible application cross-section. It thus represents an average user profile for batch processing. This job stream was coordinated with the USSR as a basis for the performance evaluation of ESER operating systems. It has been used for joint performance evaluations. The job stream consists of 50 jobs with a total of 148 job steps. Corresponding to work at the computer centers, the use of Sorting/Mixing represents a focal point with 16 jobs. In these jobs, files up to a size of 50,000 records are sorted. The job stream furthermore contains translation, linkage, and execution steps in the languages Assembler, PL/1, FORTRAN, and COBOL. Here, the execution steps represent the most significant portion of the generated load. Seventy-five percent of all user accesses take place with the access method BSAM (50 %) and QSAM (25 %).

The job stream contains about 10,000 records as SYSIN data. In the OC-6.1 EC, the SYSIN data are collected together in blocks of 3200 bytes each. In the SVS 7.0, the majority of temporary files were SPOOL files. The systems of the OC-6.1 EC and the SVS 7.0, which were used for the measurement, were generated with the same generating parameters; the arrangement of system files on the data medium was likewise comparable.

The measurements were performed under comparable external conditions. The measurement conditions were specified so that reproducible results could be achieved with small, negligible fluctuations in the values. The system files SYS1.SYSJOBQE, SYS1.PAGE, and SYS1.SPOOL were identically set up with individual measurements. The same parameter specifications were used during system start. During the measurements, all external influences (e.g. by operator activities) were avoided.

For the measurements on the OC-6.1 EC and the OC-7 EC, new program execution organizers were equally used. The job stream was read in from tape via a system input program. The system output took place on a printer. The measurement interval extended from the start of the first job to the completion of the last one.

To enter the system-internal parameters, the same measurement program was introduced into the OC-6.1 EC and the SVS 7.0. This program was specially developed for these purposes. By means of this measurement program, the following internal system parameters can be measured in addition to the start/stop times:

- waiting time
- CPU time, subdivided into SV, PPN, and PPO times
- accesses to external units
- number and processing time of supervisory calls
- number of interrupts, coded according to SVC, I/O, and program interrupts

The measurement program eliminates its intrinsic demand for CPU time. The measurement program does not create additional I/O operations during the measurement. The measurements were performed on a model EC 1055 M with a 3 Mbyte large real memory. The measurements were performed with three 100- and three 200-Mbyte WPS on separate channels. The results of comparison measurements between the OC-6.1 EC eighth modification and the SVS 7.0 are summarized in Table 8. The processing time in the SVS 7.0 could be reduced to about half of the processing time in the OC-6.1 EC SVS (eighth modification). This corresponds to a throughput increase of about 100 percent for the batch processing average profile that was used here.

Table 8. Comparison of the efficiency of the OC-6.1 EC, eighth modification SVS, and the SVS 7.0

	OC-6.1 EC SVS	SVS 7.0	relativer Aufwand im SVS 7.0 [%] 1
Gesamtlaufzeit	3452 s	1748 s	50,6
Wartezeit 2	24 s	30 s	125,0
SV-Zeit	2406 s	665 s	28,6
PPO-Zeit	152 s	171 s	112,5
PPN-Zeit	870 s	882 s	101,3
SIOs für das 3			
Betriebssystem	35 896	20 992	58,5
Residenz	17 210	7 409	43,1
Jobkette	18 686	13 583	72,7
SIOs für Dateien, einschließlich temporäre Dateien 4	78 862	36 790	46,7
Anzahl der EXCP 5	130 068	17 269	13,3
Anzahl der EXCP + EXCPVR	133 302	53 540	40,2
davon EXCPVR	2 %	68 %	
Anzahl der XCTL	40 516	1 091	2,7
Anzahl der GETMAIN	185 899	42 453	22,8
Anzahl der SVC-Unterbrechungen	630 400	233 733	37,0
Anzahl der PGM-Unterbrechungen	24 500	13 549	55,3
Anzahl der E/A-Unterbrechungen	173 229	66 934	38,6
SIOs für Eingabeband 6	12 454	820	6,7
SIOs für Drucker	19 957	1 050	5,2

1 relative expenditure in the SVS 7.0
2 total running time
waiting time
SV time
PPO time
PPN time
3 SIOs for the
operating system
resident
job chain
4 SIOs for files, including temporary files
5 number of EXCPs
number of EXCP + EXCPVR
of this EXCPVR
number of XCTL
number of GETMAIN
number of SVC interrupts
number of PGM interrupts
number of I/O interrupts
6 SIOs for the input tape
SIOs for the printer

From the results of the comparison measurements, it can be seen that the SV time, which means the CPU time for system control, has reduced to less than a third in the SVS 7.0 as compared to the SVS of the OC-6.1 EC, eighth modification.

The reduction of SV time is based on the developmental results of the SVS 7.0 which have been described under Point 3. The run through frequency of the supervisor was reduced to about 38 percent in the SVS 7.0 as compared to the OC-6.1 EC, SVS. Especially laborious supervisory services were reduced. Simultaneously with the reduction of SV time, the occupation times for channels, control units, and devices were clearly reduced, so that no overall increase of waiting time resulted. With the speed-up of the printer output and card input, the run through capability of the overall system was enhanced in such a fashion that the above-mentioned throughput doubling became possible.

The measurements were performed on a 100 and 200 Mbyte WPS technique. By using 29 Mbyte WPS technique, the same throughput increase is to be expected for the assumed average application program of batch processing.

Table of Abbreviations

AMH	Communication handler for application programs (for communication from and to application programs)
BC mode	Basic control mode
BDAM	Single direct access mode
BIU	Basic information unit
BLU	Basic linkage unit
BPAM	Simple access method for subdivided files
BPS	Operating system of the OC-7 EC
BSAM	Simple sequential access method

BSC	Line procedure for synchronous transmission of binary coded data
BTAM	Basic access method for remote data processing
CP	Control program for the SVM
CPM	Linkage manager
DDM	Dynamic test monitor, component in the OC-6.1 EC
DFC	Data flow control
DLC	Data linkage control
DMH	Communication handler for devices (for traffic from and to the data stations)
EC mode	Expanded control mode
EP	Emulation program
ESPD-RGW	Uniform system of program documentation in the member countries of the CEMA
FID	Format indicator
GTF	Tracing aids, component in the OS/ES
ISAM	Indexed sequential access method
KPTO	Complex of programs of technical maintenance
LPA	System area for resident loading modules
MCP	Communication control program in the TSO
MFT	Control program configuration for multiprogram operation with a fixed number of jobs (OS/ES, until Edition 6.1).
MH	Message handler
MMS	Multimachine support, component in the OS/ES
MVT	Control program configuration for multiprogram operation with a variable number of jobs (OS/ES, as far as Edition 6.1)
NAU	Network-address unit
NC	Network control
NCP	Network control program
OLT	Control program for device test
PC	Path control
PCP	Control program configuration for monoprogram operation (OS/ES, Edition 4)
PDAS	Error analysis system of the SVM
PIU	Path information unit
POC	Multicomputer control system of the OS/ES
PSW	Program status word
PTS	Dialogue-capable programming and testing system of the SVM
QSAM	Expanded sequential access method
RFTS	Remote file transmission system of the SVM
RU	Request/response unit, smallest SMA information unit
SC	Seat control
SDLC	Line procedure which represents an expansion of the BSC technique
SLM	System load measurement, function in the SVS
SNA/ES	Network architecture for remote data processing of ESER
SPAM	Access method for SPOOL system in the SVS (QSAM/BSAM compatible) and for managing temporary files
SPOOL system	Program to manage system input and system output
SSCP	Control of resources based on the principle of network architecture by means of a control central

SVS	Control program configuration beginning with Edition 6.1 of the OS/ES, system with a virtual address space
TC	Network control
TCAM	Expanded access method for remote data processing
TCAM/NF	Expanded access method for remote data processing with network functions
TCB	Job control block of the TSO jobs
TOTE	Control program for device test under TCAM
VSAM	Generalized access method for direct access devices

Virtual Storage Access Method

East Berlin EDV ASPEKTE in German No 3, 1984 (signed to press 30 May 84) pp 20-24

[Article by Joerg Neumann, Rainer Theile, Laus-Dieter Scheler, VEB ROBOTRON Center for Research and Engineering]

[Text] Information gained in the use of the access method VSAM of the operating system OS/ES, Edition 6.1 permitted a functional expansion to be undertaken for the OS/ES of Edition 7 for VSAM. This access method is designated in this article as VSAM2.

The objective of the article is to provide a general presentation of the most important functional expansions. Basic knowledge of the access method VSAM and experience in the use of its most important functions is presupposed.

1. Introduction

With the generalized access method for direct access units (VSAM), the Modification 4 of the OS-6.1 EC furnished an access method which is tailored to working with direct access units and which optimally uses the capabilities of virtual memory. VSAM is designed to fulfill most of the requirements imposed by batch processing and remote data processing. With this access method, sequential as well as direct accesses are possible, but the accesses can be made by means of a code or a relative byte address. Various processing types permit parallel processing in a common data base system. The user can select one or more access types, such as corresponds best to his particular application.

Likewise, it is possible to use the utility program IDCAMS and the VSAM programs under the control of subscriber support (TSO).

The access method VSAM offers a multiplicity of application variants. Thus VSAM has a data format which allows it to store data independent of the type of the direct access unit. The access method permits sequential, jump-sequential, or direct access to the data; it permits different variants for performance optimization; it has a comprehensive catalogue, in order to define storage locations and files on the direct access data media; and it offers a universal utility program by means of which the catalogue and the files can be constructed, processed, and maintained. Furthermore, this new access

method is distinguished by high power, simple utilization, expanded and easily used data protection, central control, and easy conversion of ISAM files into VSAM files, and the further processing of existing ISAM programs.

By means of the OS/ES of Edition 7, a VSAM with significantly expanded functional scope is furnished in the SVS, BPS, and PTS. This VSAM version is designated in this article as VSAM2. The access method VSAM2 offers the user the capabilities for forming and processing alternative indices, forming and processing records in number-sequence files (RRDS), using reusable files, using segmented records, utilizing sequential backwards processing, using improved control interval processing, using joint resource complexes, and using various catalogue restoration procedures. By means of these functional expansions, the application areas of the access method VSAM2 as compared to VSAM are clearly augmented, and many user requirements are fulfilled. Because of the compatability of VSAM files and VSAM catalogues between VSAM and VSAM2, it is possible to have a problem free transition to the new version of the VSAM. Likewise, the exchange of VSAM files between the operating systems SVS, BPS, and PTS is guaranteed.

2. Alternative Index

2.1 The Code Sequence File and its Restrictions

A code sequence file consists of two parts, the data component and the index component. Both the data and the index are stored in areas of fixed length, namely the control intervals. Each control interval can contain one or more records. In each data record, at a fixed defined point, there is a defining argument of constant length, called the code. The position and length of the code are fixed when defining the association for the code sequence file.

The index comprises the relation between the code and the position of the associated control interval within the data interval. A direct searching of the data record and a sequential searching according to logically increasing values of the codes is thus effectively possible. The access method VSAM implements a required access - to present it simply - in the following manner: In the index component, a corresponding index entry is determined for a given code. This index entry contains the number of the data control interval in which the record must be situated, if there exists a corresponding one.

Since all data control intervals have the same length, the access method can determine the displacement of the desired control interval with respect to the beginning of the file, the relative byte address, and thus the physical record address.

Simplified representation:

Index entry ... | LEHMANN ... 2 | ...

The entry indicates that, in the data control interval with number 2, the data component described by this index entry may contain only records whose code is not logically greater than the logical value of the character chain LEHMANN.

Data set ... ... LEHMANN ... ...

control interval 2 — record n — end of control interval

If one assumes that the control interval is 512 bytes long, this means: The record with the code LEHMANN must be searched for in the control interval whose displacement with respect to the beginning of the file is 1024. Since the physical record address of the first record is already fixed by applying the linkage, the direct read from external memory is possible.

The above-described method of managing a code-follower file entails some restrictions. All data records contain a defining argument of the same type, since the position and length of the code for all records are the same. This defining argument furthermore must be unique which means it may not repeat in any record. Thus, the handling of data in the sense of a code-follower file is sometimes impossible if the records are characterized by the same properties. If access to the data is to take place according to various logical perspectives, that is according to different defining arguments, the method described above makes necessary the multiple storage of data with a reconstruction of the index part in each case. Furthermore, each change of data requires an updating of all file copies.

Access to the data, managed via several defining arguments, also is not possible through other access methods of the operating system OS/ES.

The access method VSAM2 offers a function which eliminates these restrictions.

2.2 Treatment of a Code-Follower File by means of an Alternative Index

The following requirements are to be fulfilled by an alternative index:

- The formation of indices for an existing code-follower file without multiple storage of the data components should be facilitated. These new indices must guarantee all advantages in terms of work as the does the existing index.
- The new indices that are being formed must be capable of managing various records with the same code.

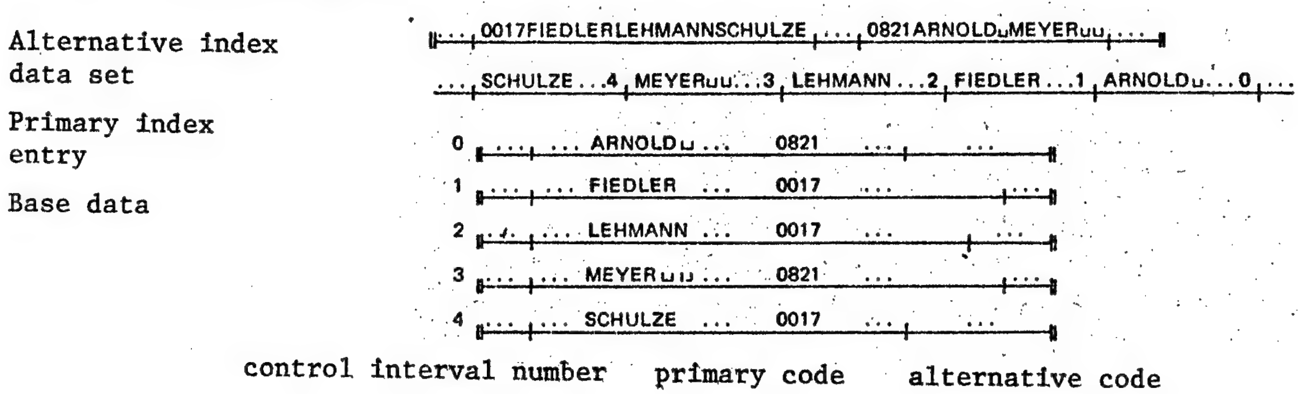
A file for which an alternative index is being formed below shall be called a base file. An alternative index corresponds in its structure to a code-follower file, consisting of an index part and a data part. The data part contains the assignment of the alternative code with the associated base code. The index part manages the data part; its format and its function are analogous to the index part of the base file.

By means of an alternative index, a base code is determined by the access method, starting from an alternative code. By means of this, the desired base data set can be localized.

By using several alternative indices relative to the same base file, the codes can overlap arbitrarily. The like holds for the superposition of base codes and alternative codes (Figure 1).

Figure 1:

Simplified representation:

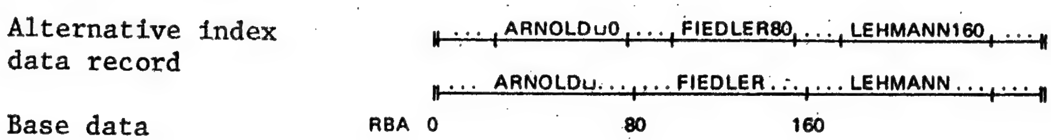


2.3 Treatment of an Access Follower File by means of an Alternative Index

With the formation of an alternative index on an access follower file, it becomes possible to refer to an access follower file in the sense of a code follower file. In this way, the data record of an access follower file can be selected according to their content. Direct access to a record without knowing the position of the record within the file is thus made possible. Likewise, the file can be handled in a contextually logical sequential sequence, depending on the choice of code as part of the data. This mode of operation is implemented by the access method in the following manner: In the alternative index data record, there is contained the relationship between the alternative code and the associated relative byte addresses of the data records of the access follower file (Figure 2).

Figure 2:

Simplified representation:



2.4 Maintenance of Alternative Indices

All alternative indices of the base file and the associated base index manage the same data set. But this means that every change of data, whether it now takes place through an alternative index or through the base index, can garble the relationships between the alternative indices and the base file. For example, if a data set is changed through the base index in such a fashion

that the alternative codes also change, without updating the associated alternative index entries, the processing of the data through the alternative indices will become erroneous. The user must therefore take care that synchronicity prevails between the base file and all associated alternative indices, if the special problem requires this. There are two possibilities for this:

When defining the linkage for the base file and when defining the associated alternative indices, the user specifies that the access method must always guarantee synchronicity. As a consequence, with each change of the base data, not only the base index, but also every alternative index is immediately updated. This mode of operation causes a reduction of processing speed.

The second possibility is that the user himself takes over the maintenance of the base index and the alternative indices, for example by regular renewed formation. This may be more efficient than synchronization performed via the access method if either there are no index changes as a result of the data changes, or if the accesses via the indices occur at fixed points in time and at larger intervals.

The answer to the question, whether the access method or the user himself should maintain the indices, thus depends essentially on economic perspectives, which again are decisively determined by the application purpose of the data set. It should further be mentioned that, along this path, different rights as regard data regulation could be assigned to several users of the same data set.

3. Record Number Follower Files

To store records of fixed length, a new form of file is being made available. The control intervals are subdivided into segments of equal length corresponding to the records. These segments are represented by a number. Exactly one number corresponds to each segment of the file, depending on the position of the segment within the file. Records can be stored in these segments. The handling of files organized in this fashion offers similar advantages as for a code follower file. To identify a record, the user can use a value that has been assigned to him without knowing the position of the record within the file. However, in contrast to the code follower file, the argument is not part of the data.

An index is not necessary for a record numbering sequence file. Thus, processing is more efficient than for code follower files. Since all records have the same length, and since the control intervals are in any case the same length, the access method can calculate the position of the desired record within the file from the record number. This makes a direct search possible.

The subdivision of the file into sequentially numbered segments of equal length is already specified in the definition of the association. Thus, the possible operations for such files are also determined. In principle, a shift of the data as a result of insertions or deletions is not possible.

Every record is bound to its position relative to the beginning of the file, in virtue of its number. However, whether a particular segment contains data or whether it is available for data storage depends on the operation associated with the segment. The deletion of records thus again leads to available segments, and the outputting of records causes a modification of the segment content. Thus, records can be inserted only into available segments.

A record number follower file can also be processed sequentially. Starting from the current record position, the data are handled in the numerical sequence of their record numbers. During sequential read, the segments which are identified as available are passed over, and the process continues with the next record.

For record number follower files, the formation of alternative indices is not permitted.

4. Segmented Records

Every VSAM file is subdivided into control areas and these again are subdivided into control intervals. The control interval generally comprises several records. The longest record up to now had to be able to be stored in one control interval. Thus, in the definition of a linkage, the control interval was essentially influenced by the maximum record length. For the access method VSAM2, the data records can be longer than one control interval for code and access follower files. The length of the control area forms the upper limit. The records are subdivided into segments by the access method. This mode of operation must be fixed when defining a linkage. Since the data records of one linkage can have different length, that is very short ones and individually very long ones may be used, the size of the control interval nevertheless can still have efficient values. A segmented record always begins at control interval boundaries and occupies more than one control interval. The control interval, in which the last segment is stored, may be partially empty. This storage space is no longer used by the access method for further records or segments.

Segmented records and nonsegmented records can be used mixed in a file for segmented records. The nonsegmented records are then stored as in files for nonsegmented records.

A typical example of using segmented records is practiced by the access method itself. For the data section of an alternative index, when using several base codes, segmented records are constructed for an alternative code.

5. Record Entry Sequential Backwards

Records or control intervals of code, access, or record number follower files can be sequentially processed. This means processed either according to ascending relative byte addresses, that is in the physical storage sequence, or for records of code and record number follower files, in ascending code sequence. The access method VSAM2 offers a new variant of sequential processing for records, but not for control intervals. Instead of the physical

or logical next record in ascending sequence, the record with the next lower relative byte address or with the next smaller code can also be requested. This sequential backwards processing is allowed for record positioning, record input, record exchange, and record deletion. The record numbers of a record number follower file are regarded as codes.

The sequentially backwards processing of code access follower files via an alternative index is likewise permitted. Here the determination of the previous record through an alternative index implies the determination of the base data record which contains the next lower alternative code. If several base data records exist with the same alternative code, they are made available in the sequence in which they are indicated in the alternative index.

6. Reusable Files

Every VSAM file is characterized not only by its physical attributes, but also by the scope of data it contains and by the operations that have been performed with the files since its definition.

When using a VSAM file, the user must know the content of the file and must absolutely take this into account, even if it is not significant for solving his problem. The access method VSAM2 here offers an improvement in the form that the user can leave out of account the current content of a VSAM file which has no alternative index, which does not have its own data space, and which is not subdivided into code follower sections. This file attribute must be fixed during the definition. Tts consequence is that such a file, when it is opened, can always be regarded as empty. It is therefore immediately usable. The property of reusability is specified when defining the associated linkage; the requirement whether or not the file is to be regarded as empty upon being opened is decided via the access method control block.

7. Improved Control Interval Access

A significant increase of processing speed for control intervals is achieved with the introduction of an improved control interval access in addition to the previously known working mode. This processing method can be requested via the access method control block. It is applicable to linkages of access follower files and record number follower files, data sections of code, access, and record number follower files and to index sections of code follower files. Only the control interval inputs in direct access via the relative byte address and the rewriting of a previously read control interval may be requested. As a special restriction with the above-mentioned operations, it should be noted that the linkages may not be empty and that the physical record length must correspond to the size of the control interval.

The high working speed when using the improved control interval access is achieved essentially as follows: Among other things, the access method presupposes that the requests are correctly coded, and only the services described above are requested. The file statistics are not kept, and no data motions are recorded via output routines. If the user violates these

requirements, the mode of operation of the access method is not foreseeable and the result is not valid. The use of this access mode is therefore usable satisfactorily only if the corresponding program has been checked for freedom of error by means of the normal control interval access.

8. Use of Joint Resource Complexes

The access method VSAM2, in Edition 7 of the OS/ES operating system, makes possible the joint use of selected resources for various files within one region, in order to reduce the scope of the necessary resources and in order to use the resources more efficiently. The totality of these jointly used resources is designated as a resource complex. In particular, this resource complex contains the buffer areas, which are necessary for reading, searching, and writing in connection with control intervals, the required channel programs, and the control blocks (input/output block and position marker block).

For using the capabilities offered by the access method VSAM2 to use jointly a resource complex, a series of macro statements are made available:

BLDVRP	construct a resource complex
DLVRP	remove a resource complex
WRTBFR	write a buffer
SCHBFR	search a buffer
MRKBFR	identify a buffer
SHOWCAT	furnish file information from the VSAM catalog

The macro statements ACB and RPL are expanded by the operands, which are needed to work with the resource complex.

By means of the SHOWCAT macro statement, information can be obtained from the VSAM catalogue concerning the files being processed. This information is needed to specify the number and size of the buffer areas. The jointly used resource complex is formed with the macro statement BLDVRP. In connection thereto, those files can be opened which use this resource complex.

The I/O buffers are handled by the access method VSAM2 in a different fashion and this can be controlled by the user to a certain extent. Unless the user requires otherwise, all write operations are delayed until

- a macro instruction WRTBFR is used or
- a free buffer is needed to implement a GET request or
- if the processing of the file is terminated with CLOSE.

In this way, the required I/O operations can be restricted to a minimum. The number of necessary write operations depends directly on the number of buffer areas utilized.

Additional functions for handling the buffer areas are made available to the user by the access method VSAM2, by means of the macro statements SCHBFR and MRKBFR.

The macro statement SCHBFR makes it possible to search the buffer for data whose relative byte addresses (RBA) lie in a prescribed range. The appropriate buffer is communicated to the user, if the desired RBA can be found.

Normally data are written into buffer areas not by a direct write but by using working areas in which the record being outputted is made available to the access method. If the buffer content is to be changed directly by the user program, an exclusive access must be requested with the macro statement MRKBFR, and this can subsequently again be reversed with this macro.

By means of the macro statement DLVRP, the resource complex that is being used can again be released, if all files which have used this complex have been terminated with CLOSE.

9. Protection and Reconstruction of Catalogues

When working with the access method VSAM, every file that is used is entered in a VSAM catalogue. The information in the catalogue describes the data media that are being used and the files contained therein. The access method VSAM takes information from the catalogue entries in order to assign storage space for the files, to check the access authorization to the files, to keep statistics concerning their use, and to establish the connection between the relative byte addresses and physical addresses. If a file entry in the catalogue now becomes useless because of a system or an I/O error, access to this file frequently is no longer possible. With the access method VSAM, reconstruction of these erroneous entries could be accomplished only with a great deal of effort by means of rescue procedures using a conventional utility program or through a current rescue status which was generated with the utility program IDCAMS by means of the function statement EXPORT or REPRO. The currency of the data was then guaranteed only rarely.

The access method VSAM2 now permits the main catalogues and the user catalogues to be defined as reconstructable. On every data medium which it manages, a reconstructable catalogue has a catalogue save area which contains copies of the catalogue entries, which are necessary to reconstruct the destroyed entries. The storage space for this information is provided automatically when the catalogue is defined or when management by the access method VSAM2 is requested for a data medium. For the save area, there is no additional entry in the VSAM catalogue. The access method VSAM stores the physical track address in the format-4 record of the data medium. This information in the catalogue save area of a data medium is always changed by the access method VSAM2 when the parallel information in the catalogue is modified.

If it is no longer possible to access entries in a reconstructable catalogue, or if it contains wrong information, it is now possible, in most application cases, to reestablish a usable status. As an aid for a reconstructable catalogue, the following functions are available: LISTCRA, EXPORTRA, IMPORTRA, and RESETCAT of the utility program IDCAMS.

The function LISTCRA is used to print out the entire content of a certain catalogue save area or to print out the entries which differ compared to those in the catalogue. By means of the LISTCRA function, an aid is made available

which facilitates the decision concerning the required correction methods for reconstructing the catalogue.

The function EXPORTRA makes copies for reconstructing objects of one or more save areas or of a data media or of a particular file. EXPORTRA makes it possible to access the data of the catalogue save area and the VSAM data, if the VSAM or non-VSAM files can no longer be addressed through the catalogue. This new function can be used in order to prepare for reconstruction non-VSAM files and individual or all VSAM files of a VSAM catalogue.

The function IMPORTRA reestablishes the catalogue information for a VSAM catalogue of non-VSAM and VSAM files, by means of the copy generated by EXPORTRA. A clear initial situation must be created to use IMPORTRA. If possible, a new or a reloaded VSAM catalogue should be used. IMPORTRA presupposes that the data medium on which the exported VSAM files exist can be made accessible.

In constrast to the functions EXPORTRA and IMPORTRA, the RESETCAT function is a one-step operation which makes it possible to reconstruct an entire VSAM catalogue without data transport. RESETCAT examines or processes no data, but compares the catalogue entries with the information in the catalogue save area. The catalogue entries from one or more catalogue save areas are mixed in a working area with the records of the catalogue that is being reconstructed. Here, those catalogue records which correspond to records of the catalogue area are replaced. The catalogue records which do not correspond to a record of the catalogue save area are then deleted, and the records of the catalogue save area to which their corresponds no record in the catalogue are added. Then the VSAM catalogue is brought to the status of this working area.

If it becomes necessary to erase VSAM data media during the reconstruction operations, the access method VSAM2 offers the possibility of erasing even a data medium with an unknown content of all VSAM data areas and resetting the VSAM ownership rights for the data medium whose catalogue entries cannot be localized. New utility program functions can be used, independent from the application, to erase a user catalogue, without first having to remove each file of the catalogue. Non-VSAM files on the data medium are not affected. The access method VSAM2 erases all VSAM data areas and writes the VTOC of the data media anew. But this function should only be used if access is no longer possible to the VSAM user catalogue which manages the data medium.

All these new reconstruction possibilities significantly increase the availability of the VSAM data. The use of these means must be tailored to the individual application and must be carefully considered. Thus, the currency of the required catalogue entry or of the entire catalogue can again be reestablished.

Assembler 2 Program

East Berlin EDV ASPEKTE in German No 3, 1984 (signed to press 30 May 84) pp 25-26

[Article by Rainer Kretzschmar, VEB ROBOTRON Center for Research and Engineering]

[Text] Assembler 2 represents a further development of the Assembler that is contained in the operating system OS/ES, Edition 6.1. An improved language translator was here created and the Assembler language was also expanded. Upwards compatability was guaranteed. The new language translator is characterized by short translation times and by a reduction of the required external storage capacity. Only one working file is now needed and this only if the main storage area furnished for translating a source module is not sufficient to receive all the necessary data. Assembler 2 involves a component of Type 2.

For the design and implementation of user programs, one question among others that must be answered, is in what language one should program. In the operating system complex OS/ES, Edition 7, the problem oriented languages PL/1, FORTRAN, COBOL and PASCAL are available for this purpose as well as the machine oriented Assembler language.

Depending on the specific application, one of these languages is selected. Here, the Assembler language is especially important as regards optimal results in terms of the translation product and translation time. For each of these languages a corresponding programming system exists on the operating system complex OS/ES, Edition 7, and this is stored on an independent distributor tape. For the Assembler language, Assembler 2 exists as a language translator. This involves the further development of the Assembler of the operating system OS/ES, Edition 6.1 as a component of the system. The transition from a component of the operating system to a component of Type 2 yields the following advantages in this case:

Independent generation of concrete variants of Assembler 2 into a functional operating system.

Adaptation of Assembler 2 to special applications:

- generation of certain selection conditions, differing from the standard
- generation of certain names for DD statements, differing from the standard
- generation of certain operation code tables in dependence on the programs being translated.

With the further development of the Assembler to the Assembler 2, the user has available not only the standard language scope but also faster language translation. This means that about 30 percent less time is needed for translating a source module with Assembler 2 as compared to translating the same module

with the previous Assembler. In principle, it is here guaranteed that programs which are written in the Assembler language will yield the same translation results with Assembler 2. The increase of efficiency in Assembler 2 as compared to the Assembler is essentially implemented by an improved organization in the processing of internal data. Thus, Assembler 2 requires only one working file compared to three for the Assembler. If adequate main storage space is available, the translation can even be managed without using the working file.

The processing of macro definitions from source text libraries was made significantly more efficient. In the Assembler, each macro call is converted into a search, a read, and an insertion of this macro definition into the source text. In Assembler 2, each macro definition is read only once, independently of how often the associated macro call is programmed. With each further call, the linkage to the already read-in macro definition is established. Macro definitions of the primary Assembler input can be programmed at an arbitrary point in the source module. The only requirement is that they are placed before the first call.

Since the structure of the source module significantly influences the main storage space that is required, only one guide value should be given for a possible REGION specification for the translation, as a reference point. For a source module which consists of about 1500 statements of primary Assembler input, where an additional approximate 4500 statements are read in from a source text library (especially macros), a specification of REGION = 700K is sufficient for the working file not being needed. As a minimum main storage requirement for translating a source module, one needs 280 Kbytes.

The preparation and output of error messages was also improved compared to the Assembler. Thus the error cause in an erroneous statement is shown immediately following this statement in the Assembler protocol. Connected with this is a specific formulation of the error cause and a reference to the erroneous part of the statement.

Example:

```
                                       .
                                       .
                                       .
                PUNCH 'MSGEDIT  ((RA.,X'37')), BLANK=NO'
IUM63***ERROR*** OPERAND NOT PROPERLY ENCLOSED IN QUOTES
                                       .
                                       .
                                       .
```

The error message says that, in the operand of the Assembler statement PUNCH, an unpaired apostrophe occurs (at x'37'), but that this is not permitted.

For special applications - translation of several source modules with the same source libraries as well as sequential output of the translation results - these source modules can be translated in one translation run. This reduces the effort for initializing several job steps to initializing a single job step. Another increase of efficiency can be achieved if the offered language extensions are utilized.

These concern mainly the language elements of the necessary translation and the complexity of the statements. New language elements supplement the transfer of values between primary Assembler input and macro definitions, support the error search in macro definitions (macro trace function), and permit an expanded structurization of the source module. For improved symbolic programs, the allowable number for the formation of symbols was expanded from eight characters to a maximum of 63 characters.

The following example demonstrates these language extensions:

```
LCLA &STANDARDWERT,  1                             X
         &ANFANGSWERT,&ZEIGER(2)
              .   2          3
              .
              .    1                   5
AIF      (&STANDARDWERT EQ 1).ZYKLUSSTART1,        X
         ('&ZEIGER(4)'       EQ '1').ENDANWEISUNG
              3                          4
              .
              .
.ZYKLUSSTART1       LA      REGISTER12,HAUPTSPEICHER-
       5            .                    6     ADRESSE
                    .
                    .
.ENDANWEISUNG       BR      REGISTER14
       4            .
                    .
                    .
HAUPTSPEICHERADRESSE DS       CL121
       6            .
                    .
                    .
                    END
```

1 standard value
2 initial value
3 pointer
4 end statement
5 cycle start
6 main storage address

A few of the language extensions will be listed here, since using them makes the translation time even more efficient. Thus the definition of local SET symbols can be omitted by means of LCL statements. The values are assigned during the first use of such symbols in SET statements. Global SET symbols must continue to be defined as such by means of GBL statements; however, the position of these statements both in the source text as well as in the macro definitions is arbitrary. As a restriction, it should be noted that the definition of global SET symbols must take place before they are used. The functional extension for SET statements can be used to assign values to SET symbols. Thus it is possible, by means of one SET statement, to assign values to several SET variables (only indexed SET symobls are allowed here). Here the dimensions of an SET symbol can be expanded at any time during the generation.

Example:

```
&SETA(4)        SETA     6,7+&SETA(3),,,,9,11
```

The SETA variables &SETA(4),...,&SETA(10) have new values assigned to them by means of an SETA statement. Here the values for &SETA(6), &SETA(7), and &SETA(8) remain unchanged, if values were already assigned to them, or are set to zero if this is the first value assignment.

From this presentation, it is clear that the number of statements to be processed has been reduced. In the language of the Assembler, six SETA statements would have to be written for this.

A language extension was also implemented with the statements AIF and AGO. By means of such an instruction, it is now possible to interrogate several conditions and to transfer to various statements that are identified by sequence numbers.

Example:

```
AIF (&A EQ 4).Z1.(&BETA(7) NE &C(2)).Z2
```

Depending on the truth values of the two logical expressions, each of the statements is processed which is designated with the sequence number .Z1 or with the sequence number .Z2. The expressions are here evaluated from left to right; the transfer takes place if the first expression has the logical value "TRUE".

Example:

```
AGO (&WERT).GO1,.GO2,.GO3
```

The arithmetic value of the variable symbol &WERT yields the transfer condition. If this value lies between 1 and 3, the transfer occurs to the statement designated by the sequence numbers .GO1, .GO2, .GO3; otherwise, the AGO statement is processed in the following fashion. A feature associated with the extension of the function of the statements AIF, AGO, SETA, SETB, and SETC, also is an increase in the number of continuation lines to nine, as well as the representation of these statements in the "expanded" statement format. Thus, the AGO statement can also be coded as follows:

```
AGO  (&WERT).GO1                    X
     .GO2,                          X
     .GO3
```

From what has been presented here, it is clear that the translation time of the Assembler 2 has been shortened compared to the Assembler, by using all the language extensions and if the main memory area for translating a source module is sufficinetly large.

Assembler 2 no longer supports the following selection conditions: LIBMAC, MLOGIC, NUMBER, STMT, MCALL and ALOGIC. Another possibility was created for

this, to facilitate an error search in macro definitions, especially involving those from a source tape library. By means of the instruction MHELP (macro trace function), it is possible to record the generation process for macro definitions involved in a macro call. Depending on a specified parameter, the values of variable symbols can be indicated at particular times during the generation.

The Assembler 2 can be used in the operating systems SVS, BPS, and SVM of the operating system complex OS/ES, Edition 7, in the operating system OS/ES, Edition 6.1 beginning in modification 8, with the TSO MVT 1.4 and the SVS 1.1, as well as in the SVM/ES beginning with Edition 1.2. The above-mentioned operating systems are generated by means of a distribution tape for Assembler 2. The only precondition is a functional operating system. The distribution tape furnishes all the necessary aids for generating an Assembler 2.

TSO Subscriber Support Operation

East Berlin EDV ASPEKTE in German No 3, 1984 (signed to press 30 May 1984) pp 27-31

[Article by Guenter Brusdeylins, VEB ROBOTRON Center for Research and Engineering]

[Text] This article shows how the subscriber support operation (TSO) is subsumed as a component of the operating system SVS in OC-7 EC. The structure and mode of operation are explained briefly. The usable programming languages and the possibilities of user programs in TSO operation are presented.

At the end a performance comparison is given between the TSO of the SVS of the OC-7 EC and the TSO in the SVS of the OC-6.1 EC.

1. General Considerations

The operating system SVS of the OC-7 EC makes possible not only batch processing, but also the use of the data processing system (EDP system) in subscriber operation.

The TSO component is furnished as a means of implementing convenient subscriber operation.

Subscriber support is an additional function of the operating system SVS, a function which permits the simultaneous and mutually independent utilization of the capabilities of this operating system from the data station.

In practical applications, the following basic differences exist in batch processing and subscriber operation (TSO operation):

In batch operation, jobs are executed for which all the necessary data are available before processing begins. A job is given to the computer center. While this job is being processed, the original data has no influence on the course of processing.

Batch processing permits efficient work load of the EDP system, resulting in low processing costs. However, the run-through time of the job is relatively large and is essentially determined by organizational running sequences, such as job preparation, processing time, and distribution of the results.

These organizational influences are considerably reduced in TSO operation. The subscriber is directly connected with the EDP system via a data station. By means of commands, he causes his job to be processed directly and he himself enters the data. Important processing results can be obtained at the data station.

TSO operation is primarily oriented towards high efficiency in the use of an EDP system.

TSO can be used for the following tasks:

- Developing and testing programs in dialogue (dialogue programming) by using various programming languages.
- Construction and maintenance of files, which contain source text, data, job control statements, or commands.
- The processing of dialogue-capable programs, which the user has developed possibly by using the macro instructions and service routines furnished by the TSO.
- The programs can belong to various application areas, such as economy, production control, or information retrieval. These tasks are practically becoming more and more important in the use of TSO.
- Transfer of jobs to batch processing from a data station, and displaying the results for evaluation at the data station (dialogue job input).
- Processing programs within batch processing, e.g. utility programs and system auxiliary programs in the operating system SVS, and receiving the results at the data station.
- Text processing with the preparation of text files and output of the formatted text.

The TSO of the SVS of the OC-7 EC is based on the edition TSO SVS 1.1 of the OC-6.1 EC.

2. TSO Control Program

For message traffic from and to the data stations, TSO uses the expanded access method for remote data processing (TCAM/NF). For this reason, a message control program (MCP) must be generated which is suitable for TSO operation. For this purpose TSO has available macro statements. The following components belong to TSO:

- TSO control program
- monitor program
- command processor
- service routine.

The TSO control program executes jobs which are connected with the control of TSO operation. This comprises the following functions:

- checking the subscribers for their authorization to work in TSO operations
- initializing the TSO jobs
- calculating and assigning time slices for TSO jobs
- controlling the job exchange
- deactivating and activating TSO jobs before and after the exchange.

The TSO control program comprises the components:

- TSO control task
- regional control task
- TSO interface program and TSO coordinator
- TSO dispatcher
- I/O coordinator
- subscriber diponent.

Essential parts of the TSO control program work in the TSO control region.

The monitor program and the command processor are executed in several TSO regions.

Figure 1 shows the components of the TSO and their cooperation with other components.

The structurization of the virtual memories during TSO operation is shown in Figure 2.

2.1 Starting and Stopping TSO Operation

The TSO operation can be taken up by the operator at an arbitrary point in time as follows:

After the data stations and lines have been made ready, a START command is entered for the message control program. Then another START command must be entered with the name of the start procedure of the TSO control program.

TSO operation is terminated by means of the STOP command for the TSO control program. After this, the message control program can be stopped with a HALT command. TSO regions corresponding to the number and size specified in the TSO parameters are set up by the TSO control program. The TSO system parameters are situated in the inventory IKJPRM00 of the subdivided file SYS1. PARMLIB.

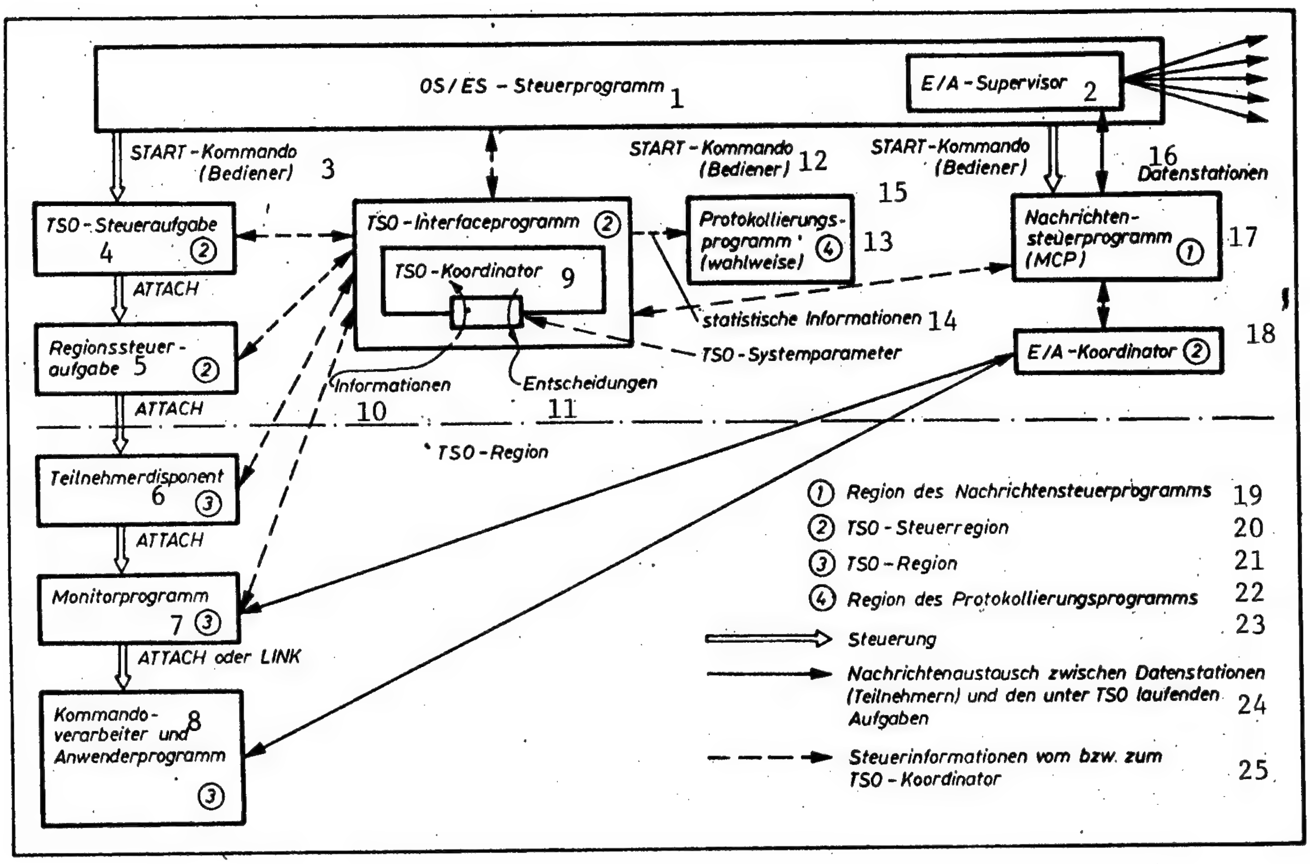
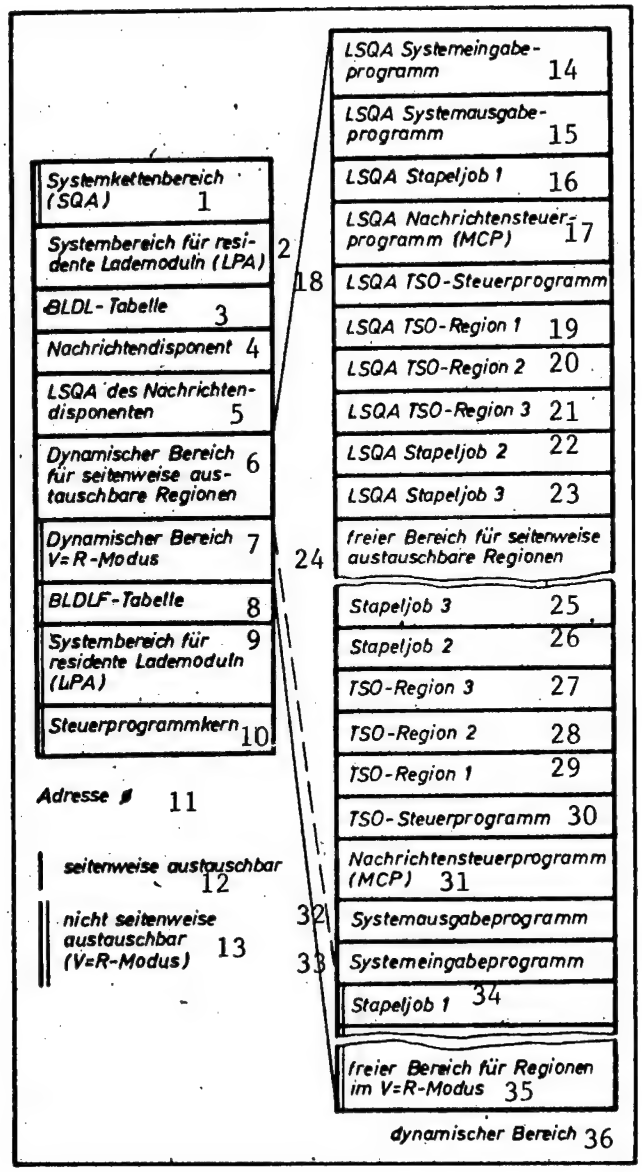


Figure 1: Schematic mode of operation of TSO

1 OS/ES control program
2 I/O supervisor
3 start command (operator)
4 TSO control task
5 regional control task
6 subscriber disponent
7 monitor program
8 command processor and application program
9 TSO coordinator
10 information
11 decisions
12 start command (operator)
13 protocol program (optional)
14 statistical information
15 start command (operator)
16 data station
17 message control program (MCP)
18 I/O coordinator
19 region of the message control program
20 TSO control region
21 TSO region
22 region of the protocol program
23 control
24 message exchange between data stations (subscribers) and the jobs running in the TSO
25 control information from and to the TSO coordinator

Figure 2. Typical organization of the virtual memory during TSO operation with batch processing



1 system chain area (SQA)
2 system area for resident loading modules (LPA)
3 BLDL table
4 message disponent
5 LSQA of the message disponent
6 dynamic area for regions that can be exchanged page by page
7 dynamic area V = R-mode
8 BLDLF table
9 system area for resident loading modules (LPA)
10 control program for
11 address Ø
12 exchangeable page by page
13 not exchangeable page by page (V = R- mode)
14 LSQA system input program
15 LSQA system output program
16 LSQA batch job 1
17 LSQA message control program (MCP)
18 LSQA TSO control program
19 LSQA TSO region 1
20 LSQA TSO region 2
21 LSQA TSO region 3
22 LSQA batch job 2
23 LSQA batch job 3
24 free area for regions that can be exchanged page by page
25 batch job 3
26 batch job 4
27 TSO region 3
28 TSO region 2
29 TSO region 1
30 TSO control program
31 message control program (MCP)
32 system output program
33 system input program
34 batch job 1
35 free area for regions in the V = R mode
36 dynamic area

2.2 Initializing the TSO Job

After the TSO has started, the linkage between the subscriber and the data processing system can be established by entering the command LOGON at a data station. In the LOGON command, the subscriber must specify the name of a LOGON procedure. The LOGON procedure is a component of the subdivided file, whose name is specified in the TSO starting procedure. It contains an EXEC statement which specifies the name of a monitor program, and also DD statements which describe the files required by the subscriber. The DD statements an also be specified here with the DYNAM operand. At a later time, these are used dynamically by the subscriber by means of the ALLOCATE command, in order to allocate further files.

The TSO control program assigns a TSO region to the subscriber and starts the TSO job described by the LOGON procedure. In this way, the monitor program receives control and waits for the input of commands from the subscriber. If more subscribers open a session than there exist TSO regions, several TSO jobs are processed in one region. However, at any one time, only one TSO job is situated in one TSO region. This is implemented by the TSO control program through the job exchange.

2.3 Job Exchange

Each TSO job is situated in its assigned TSO region for a short time, the main memory time slice (HS time slice). During this time, it can use the system resources jointly with all other jobs of the system. After its HS time slice has expired, the active content of the TSO region (the working set) is transferred into the page file. Then the TSO region is completely empty.

The active content of the TSO region consists of the pages of the TSO jobs which are currently in real memory (including pages and segment tables). Then the job set of the TSO job is transferred into the real memory, and the next HS time slice is assigned. If this job set consists of more than 16 pages (64 Kbytes), the control is already transferred to the TSO job after 16 pages have been transferred. During the HS time slice, the TSO job is subject to the normal page change of the SVS.

The process of transferring the job set of TSO jobs from the real memory into the page file and vice versa is called job exchange. The transfer of the job set into the page file is called roll-out, and the transfer of the job set into the real memory is called roll-in. The exchange is initialized by the TSO control program and is executed by the page supervisor of the SVS. Thus, the function for exchange in the TSO control program is lacking, although it is present in the TSO MVT. Before th job set of a TSO job can be rolled-out from real memory, every activity associated with it must be terminated. This process is called deactivating the TSO job. The reestablishment of activity, after the job set has been rolled into the real memory is called activation.

If sufficient TSO regions exist, so that only one TSO job is situated in each TSO region, job exchange is obviated, as well as the system load associated therewith. Furthermore, the response time behavior improves. The operating system SVS with a virtual address spaceof 60 Mbytes makes it possible to use many TSO regions (maximum 42) to strive for this mode of operation.

2.4 Subdivision of CPU Time

The TSO job which is situated in a TSO region is competing for CPU time. Furthermore, all these TSO jobs are competing for CPU time with regions in which batch jobs are being executed. For this reason, a method is necessary for subdividing the CPU time.

All TSO jobs have the same selection priority. This is guided according to the priority of the TSO control task and is generally higher than the priority of the batch jobs.

Nevertheless, there exists a priority sequence of TSO jobs. The task control block (TCB) of the TSO jobs are brought into a definite sequence by the TSO dispatcher.

Every TSO job is placed for a certain time, the CPU time slice, at the top of the TCB chain of TSO jobs, and thus receives preferential CPU time.

The magnitude of this CPU time slice is calculated by the TSO control program according to various procedures, which are specified by the TSO system parameters.

There is a simple, a uniform, and a weighted subdivision of CPU time. In the case of a simple subdivision, the entire CPU time is assigned to the TCB at the beginning of the TSO-TCB group. In the case of a uniform subdivision, the available CPU time is subdivided into equal parts per TSO job and the sequence of TCBs at the top for a CPU time slice is chained cyclically. In the case of a weighted subdivision, the magnitude of the CPU time slice depends on the waiting time percentage of the TSO jobs, which means I/O-intensive TSO jobs receive a larger time slice than CPU-intensive ones.

In general, the batch jobs must make do with the CPU time which cannot be used by the TSO jobs.

However, it is possible to guarantee a certain percentage of available CPU time for batch operation. The BACKGROUND operand of the TSO system parameters is used for this purpose. The CPU distribution to individual batch jobs is not controlled by the TSO.

3. Application of the TSO

3.1 System Availability

TSO is a component of Type 2 and is made available on its own distribution tape. Inclusion of the TSO in the operating system is effected upon a complete operating system generation.

By means of TSO, the following data stations are currently being supported: EC 7920.M (near and remote), EC 7925.M, EC 8533, EC 8565, EC 8577, EC 8562, and EC 8564.

3.3* Command Language

The means of communication between the monitor program and the subscriber is the TSO command language. By entering a command, the subscriber can request a particular function from his data station. If the function cannot be executed, the subscriber is informed with a message. If a command has been entered erroneously, the subscriber receives a request message to correct his entry.

---
*sic

The TSO commands are executed by command processors. These are load modules that can be used in parallel, and which are situated in the command library (SYS1.CMDLIB). The command processors can be made resident in the area for TSO-specific load modules (TSLPA). In this way, the loading process for frequently used load modules is avoided and thus system performance is increased.

A large number of commands for the following functions belong to the standard scope of the TSO command language:

- communication and session control (opening, monitoring, and terminating a session)
- data management (working with files)
- program development and execution (translating, testing, linking, and executing programs)
- working with TSO batch jobs (transfer of jobs from the data station to batch processing)
- TSO control (monitoring and modifying TSO operations).

3.4 Command Procedures

To simplify work at the data station, TSO offers the capability of combining frequently used command sequences into command procedures. These are stored as items in a subdivided file.

The capabilities of the command procedure are considerably expanded in TSO of the SCS OC-7 EC as compard to the TSO SVS 1.1.

3.5 Programming Languages in TSO

The Assembler and all compilers for higher programming languages, which are implemented in the operating system, can be called through the CALL command. All the necessary files must previously be allocated by the ALLOCATE commands.

For some compilers, there are intermediaries. An intermediary performs the preliminary work, for example the dynamic allocation of required files. It then calls the compiler. To develop programs more effectively, test aids are made available by some compilers (Table 1).

The intermediary and test aids of the higher programming languages are not components of the TSO.

Table 1. Survey of the intermediaries and test aids of higher programming languages

Compiler	TSO Command
PL1-OC	PL1
PL1-TC	PL1C
Programming system COBOL	
COBOL compiler	COBOL
COBOL dialogue test	TESTCOB

Table 1 (continued)

Compiler	TSO Command
Programming System FORTRAN	
FORTRAN-SE	FORTSE
FORTRAN-CC	FORTCC
FORTRAN dialogue test	TEST FORT
FORTRAN conversion program	CONVERT

3.6 Application Programs in TSO Operation

3.6.1 Programs for batch processing

TSO treats the data station like a sequential file. This means that the data station can be used as an I/O unit for the sequential access methods (BSAM, QSAM).

This makes dialogue possible in a simple manner, for example if an input file for control cards as well as the output of a printed list is allocated to the data station (by means of the ALLOCATE command). This mode of operation can be used, for example, to process the utility programs of the OS/ES in TSO operation.

3.6.2 User dialogue programs

For the development of user dialogue programs for TSO operation, three methods can be used for input and output via the data stations:

Macro statements of the I/O coordinator

If the program being generated is intended only for TSO operation, the input and output can take place through the data station by means of the macro statements TGET and TPUT. These macros permit line-by-line output on all supported data stations and an output in the video screen mode (total screen mode) for the video screen system EC 7920.M. Here, the capabilities of this data station can be used fully (e.g. form sheet technique). The control of the screen in this case must be performed by the user program.

Macro statements of the service routines

If a user program is to expand the command language, it is suitable to use the macro statements GETLINE, PUTLINE, and PUTGET of the service routines. By means of the macro statement GETLINE, data lines can be requested from a data station or from a main memory list. A main memory list is constructed, for example, when calling a command procedure, and is then processed. It contains the commands that have been assembled within the command procedure. The macro statement PUTLINE makes it possible to output multi-stage messages and to compose messages from several segments. By means of the macro statement PUTGET, a line can be outputted and an input line can be requested immediately (e.g. a response or the correction to an erroneous input). These

macro statements can be used only in TSO operation.

Macro instructions of sequential access methods.

If a user program is to work optionally in TSO operation or in batch processing, the macro statements of the sequential access methods can be used for input and output via the data station.

3.7 Performance comparison

The TSO of the SVS OC-7 EC has the following advantages as compard with the TSO of the SVS OC-6.1 EC:

- Better response time behavior or more subscribers for the same response times.
  The response times are shorter than in the MVT of the OC-6.1 EC.
- Improved functional scope, especially in the establishment of command procedures.

Thus, TSO becmes a still more effective means for improving efficiency.

References

/1/ Brusdeylins, G.; Werner, H.: Experience in the use of dialogue programming. rechentechnik/datenverarbeitung (Computer Technology/Data Processing), Supplementary Issue 1, 1981, p. 45.

TCAM/NF Extended Access Method

East Berlin EDV ASPEKTE in German No 3, 1984 (signed to press 30 May 1984) pp 31-39

[Article by Hans-Juergen Baumhaekel, VEB ROBOTRON Center for Research and Engineering]

[Text] The extended access method TCAM/NF for remote data processing with network functions is characterized by the support of resources of remote system and network processing of the ESER in a uniform remote data processing system that is based on TCAM/NF. It is made available in the operating system complex OS/ES Edition 7 for the operating systems SVS and BPS as a successor to the component TCAM of the operating system OC-6.1 EC. In this paper, the basic modes of operation and functions of TCAM/NF are presented, as well as the individual types of network resources that are supported. To understand this, the working principles of the EC 837X processors for remote data processing, and their interaction with TCAM/NF are considered, and an introduction is given to the network architecture of the ESER under perspectives involving applications and access methods.

1. Incorporation within the Operating System OS/ES

Within the framework of the operating system complex OS/ES, Edition 7, the component TCAM/NF is made available for the operating systems SVS and BPS to support the system and network architecture of remote data processing (DFV) of the ESER. It is an integral component for the operating systems SVS and BPS and accordingly for both systems is composed of the same module and macro inventory. It is included as a matter of standard in BPS and is included in SVS upon system generation by specifying the specification TCAM in the operand ACSMETH= of the generation macro DATMGT in the generated system. In the operating system complex OS/ES, Edition 7, TCAM/NF replaces the component TCAM which is contained in the operating system OS/ES, Edition 6.1. The basic mode of operation of TCAM/NF and its basic function correspond to those of TCAM. The scope and meaning of additional new functions and the associated extension and conversion of internal routines led to its development as a new component.

2. Basic Functions and Mode of Operation

The characteristic function and outstanding significance of the component as a whole appears in the possibility of using the resources of remote system data processing as well as resources of the network architecture of remote data processing of ESER (SNA/ES) in a uniform DFV system that is based on TCAM/NF. In this system, processing requests from various resources of the remote data processing network and of the TCAM/NF application programs can be supplied to one and the same application program which is implementing the required processing. As the inverse of this process, TCAM/NF makes possible the efficient solution of extensive information distribution problems up to information exchange between the various network resources without the inclusiong of user programs. On the basis of a DFV system which is implemented with TCAM/NF, already installed resources which are associated with remote processing of system data can be used further. These include, for example, existing TCAM user programs and user-related system solutions. They can also be expanded step by step, depending on user needs, to include the new functions and device capabilities. For this purpose, TCAM/NF is compatible with TCAM at the level of retranslating the programs which contain TCAM macros.

The support of the system and network architecture for remote data processing is expressed in the operating capability of data stations which work according to various data transmission procedures, formats, and protocols, under the control of TCAM/NF, as well as to the type of their linkage to the data processing system (EDP system).

In particular, the following possibilities must be distinguished:

A Resources of the system architecture

A1 local data stations, connected directly to the channel of the EDP system e.g. EC 7927 via EC 7922

A2 data stations which work with asynchronous (start/stop, SS) or synchronous (BSC) data transmission procedures, e.g. EC 7920, EC 7925, EC 8564, EC 8565 - SS/BSC

A21 connected with the EDP system through a DFV control unit of type EC 840X, e.g. EC 8404

A22 connected with the EDP system through a DFV processor of type EC 837X, in which an emulation program (EP) is operated, which emulates a DFV control unit of type EC 840X (EP mode of the DFV processor, EC 837X/EP), e.g. EC 8371 (VRB), EC 8371.01 (VRP).

B Resources of the network architecture

connected with the EDP system through a DFV processor of type EC 837X, in which a network control program (NCP) is operating (NCP mode of the DFV processor, EC 837X/NCP)

B1 particular data stations, which operate with data transmission procedures start/stop or BSC, e.g. EC 7920, EC 7925, EC 8565 - BSC

B2 data stations which operate according to the formats and protocols of the network architecture of the ESER, including their synchronous transmission control (SDLC).

A survey of the resources of a TCAM/NF - DFV network is shown in Figure 1. From this it is also apparent that, by means of TCAM/NF, hierarchical network structures are supported first of all, which are connected with the central computer via multiplexers or equivalent DFV processors in the emulation mode and via local DFV processors in the network control mode. In the first stage of the development of the OC-7 EC, the resources associated with EC 8404/EC 837 X-EP are supported here. In the second stage, the resources operated through EC 837 X/NCP will be added to this.

To understand the network resources better, some considerations concerning the DFV processor and its mode of operation are necessary at this point.

An essential element for implementing the network architecture of the ESER is a remote data processor of the type EC 837 X/NCP. The joint operation of TCAM/NF in the EDP system and of the network control program in the DFV processor is a precondition for implementing the principles of the ESER network architecture. Here, the NCP takes over the line and message control functions as well as certain basic functions as regards message processing. With conventional technology and modes of procedure these would be implemented by the access method. From this displacement of functions, one obtains the result that the DFV processor is the starting point for the transmission line and these are no longer conducted in the form of subchannels into the EDP system. TCAM/NF transmits and receives messages no longer directed from a data station. Rather, information is exchanged between TCAM/NF and the NCP. This takes place over a single channel connection between the EDP system and the DFV processor. Here, TCAM/NF controls the NCP, and the latter again controls the resources connected with it. These resources are consequently also designated as NCP resources. In principle, a distinction must be made

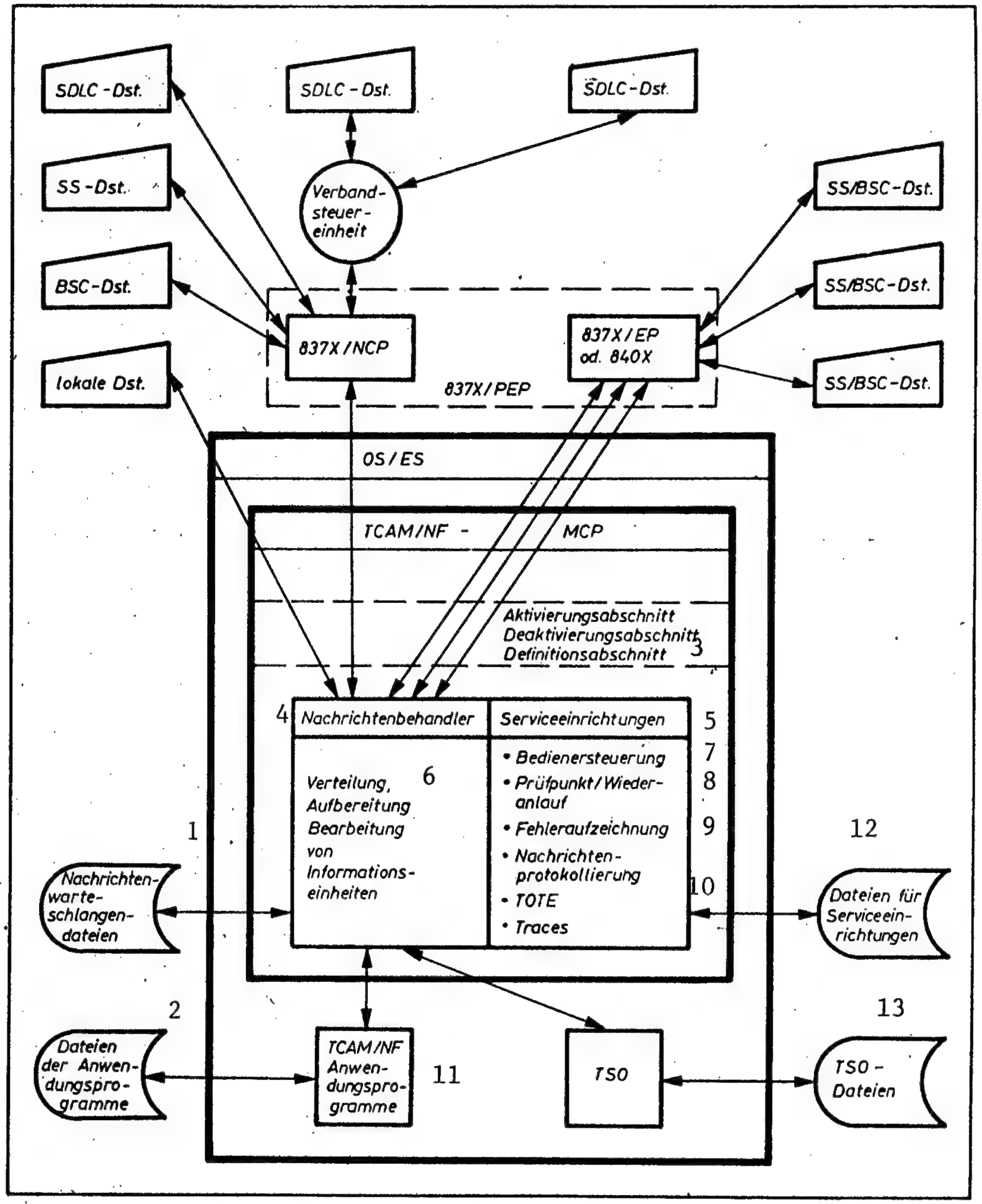


Figure 1. Survey of the resources in the TCAM/NF remote processing network

1	message queue file
2	files of the user programs
3	activation section, deactivation section, definition section
4	message handler
5	service facility
6	distribution, preparation and processing of information unit
7	operator control
8	test point/reset
9	error display
10	message protocols
11	TCAM/NF user programs
12	files for service facilities
13	TSO files

here between SS/BSC resources and SNA resources.

The specifics of the interaction of TCAM/NF and NCP require that both are coordinated with respect to one another during generation. However, the NCP is not a component of the operating system OS/ES. In particular, the NCP performs the following functions, among others:

- line control, setting up the physical linkages at the data stations, including calling, addressing, selection, responding, identification exchange
- dynamic buffer allocation and transmission of data between buffers and lines
- cotransformation for transmission to SS/BSC devices
- duplex/semiduplex control for SDLC lines
- routing of information units to the NCP resources
- data stream control on SDLC lines
- the maintenance of error statistics, error treatment
- control functions for SS/BSC units, block handling
- control functions for certain SNA units which have restricted intelligence (SNA units type PU.T1, see Section 3)
- various service functions in connection with the access method, among others line trace, test point/restart, diagnostic support, change of operation parameters.

In contrast to the mode of operation with EC 837X/NCP, TCAM/NF communicates directly with the data stations in its operating mode with EC 837X/EP and EC 840 X. Here, each transmission line has its own subchannel address. All line and message control functions are executed in the EDP system by TCAM/NF. There are no differences in the TCAM/NF treatment of lines which are connected with EC 840 X or EC 837 X/EP. The SS/BSC resources that are operated over these line connections are also designated as non-NCP resources or as EP resources.

The emulation program likewise is not a component of the operating system, but is delivered together with the DFV processor. It implements the well-known task of the DFV control units, including among others:

- matching of the channel interface to the transmission algorithm of the line
- control character identification, error detection
- time monitoring (time out)
- record testing (LRC, CRC).

The NCP can be generated so that it contains the EP. Then it is possible to operate EP and NCP lines in parallel over one DFV processor.

This mode of operation of the DFV processor is called the PEP mode.

As with TCAM, so also with TCAM/NF there is a separation of the line and message control in the TCAM/NF message control program (NCP) from the actual message processing in TCAM/NF application programs (AP). Here, messages are

exchanged between the two sides through a sequential interface. This interface is based on the use of macros connected with data management of the OS/ES (GET/PUT, READ/WRITE level) in the TCAM/NF application program.

This mode of procedure is made possible by the queue technique that is consistently implemented in TCAM/NF. For the user, this brings the advantage that he can perform work for the TCAM/NF system programmer with respect to the NCP and work for the design and programming of desired applications of TCAM/NF, in parallel and in a fashion that is largely independent of the interface relationships that are prescribed by the user. The possibilities of the network architecture and their implementation in TCAM/NF underscore this mode of procedure. The following tasks are implemented by the message control program:

- Control of resources based on the principle of network architecture, by means of a control central (SSCP)
- Organization of all operations necessary with respect to I/O operations to resources in the DFV network in the central computer, in dependence on the type of data station, the type of its linkage to the EDP system, as well as in dependence on its transmission procedure
- Queue management
- Buffer organization
- Making available information units for data stations and user programs
- Cotransformation for SS and BSC units
- Operator control to control TCAM/NF and the DFV configuration
- Special forms of message preparation
- Error handling and error recording
- Organization of test points/restart for MCP and DFV processor EC 837 X/NCP as well as coordinating the restart of TCAM/NF user programs
- Functions to support on-line tests for DFV units and lines through a device for on-line device test (TOTE), which implements a connection to the test sections (OLT) of KPTO and which thus makes possible the performance of the following tests in parallel to the normal operation of the MCP:
  - test of SS/BSC data stations at EC 840 X and EC 837 X/EP
  - test of SS/BSC data stations at EC 837 X/NCP
  - echo test for certain SNA/ES data stations
  - test of SDLC connections
- Service functions
  - message recording for message handlers
  - I/O channel interrupt trace for channels to EC 840 X and EC 837 X
  - trace for NCP control lines
  - trace for information exchange between TCAM/NF and NCP (PIU trace)
  - buffer trace and TCAM/NF dispatcher trace
  - formatized printout of disk, queue, protocol and trace files
  - formatting of the disk queue file
- Interface to the component subscriber support (TSO) of the OS/ES to implement message communication with data stations.

In the message control program, all the necessary definitions are made by the user, by means of TCAM/NF macros, for the resources of the network, such as data stations and user programs, for the buffers and queues, as well as for routines of the MCP. For special handling of information units, which refer to line groups or resources, and which are function-dependent, TCAM/NF uses a message handler (MH), which is likewise coded by the user by means of TCAM/NF macros.

A message handler investigates and processes control information in the information units and executes functions for preparing the information, for the purpose of conveying it to its destination or for the purpose of influencing parameters of the TCAM/NF system.

Two types of message handlers are distinguished:

- message handler for devices (DMH) (for traffic from and to data stations)
- message handler for user programs (AMH) (for traffic from and to user programs).

The meaning of these message handlers for the TCAM/NF message flow is shown in Figure 2.

Besides these message handlers, two further message handlers are needed for the MCP, inasmuch as the user configures the NCP resources.

The control central required for controlling these resources requires a special SSCP-MH, through which all information flows to the SSCP. If the function TOTE is furthermore requested for NCP resources, a special TOTE-MH must be included in the MCP. Both MHs can be specified in standardized form through a TCAM/NF macro.

3. TCAM/NF and ESER Network Architecture

The network architecture of the ESER formally defines the functional responsibilities of the individual components of the communication system. It provides a description of the logical structure of a DFV network as well as formats and protocols of the information flow. The architecture is implemented by means of devices, programs, and transmission procedures.

The access method TCAM/NF represents one of these means. The following considerations provide an introduction under perspectives related to applications and access methods.

3.1 Structure of the Network

The network architecture distinguishes three types of nodes, which are connected together in a network through transmission lines (channel or line):

- host nodes
  central computer (host) with operating system including DFV access method and user programs for the central control of the network associated with the host

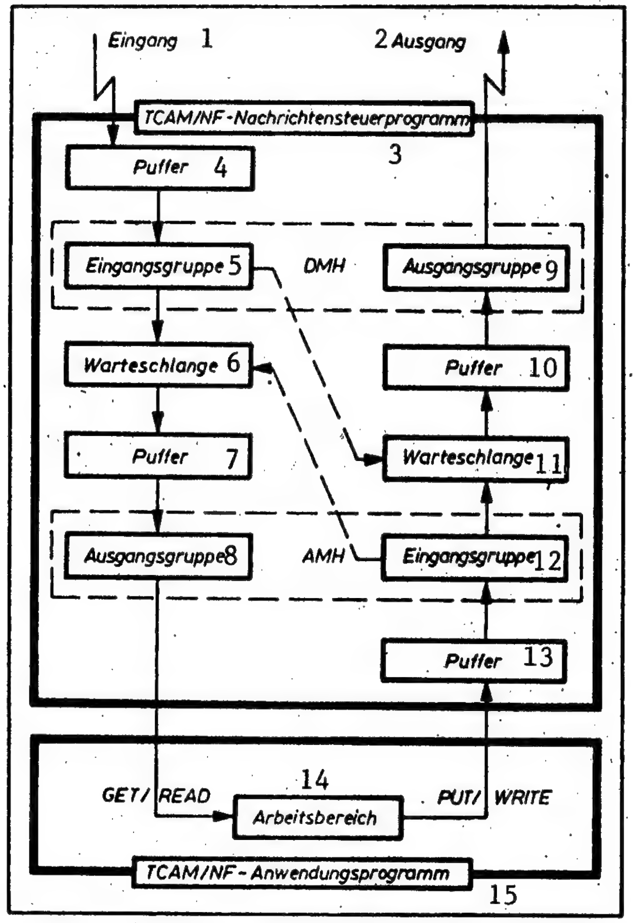


Figure 2. TCAM/NF message flow

1	input
2	output
3	TCAM/NF message control program
4	buffer
5	input group
6	queue
7	buffer
8	output group
9	output group
10	buffer
11	queue
12	input group
13	buffer
14	working area
15	TCAM/NF user program

- NCP nodes
  remote data processing processor with network control program
- peripheral nodes
  - association nodes
    association control unit with data station
  - data station nodes
    individual data stations which are not connected to an association control unit

Host nodes, NCP nodes which work together with the host node via a channel connection (local NCP nodes), as well as the network resources that are operated at these NCP nodes via line connections, together yield an individual area network. An area can contain several subareas. A subarea always is a portion of the network which is formed from host nodes or NCP nodes with the associated peripheral nodes, and which is characterized by the same subarea address. The subarea addresses are specified by the user during the generation of MCP and NCP.

There are peripheral connections between the subregion nodes and peripheral nodes, and there are subregion connections between the subregions. The subregion is composed of several elements which are implemented in the respective nodes. These elements are called network-addressable units (NAU). Three types of network-addressable units are distinguished:

- control central (SSCP)
  A central control element of a single area network, implemented in the host node, provides support for physical and logical units through a large number of command processors
- physical unit (PU)
  Control elements for all independent resources of a node, which means one PU each per host, NCP, association, and data station node.
- logical unit (LU)
  Application-oriented control elements in host, association, and data station nodes for access of the final user to the communication system.

These network-addressable units are the origin and destination of information units; every information flow always takes place between two NAUs. They are generally implemented by special programs and have a unique entering address composed of a subregion address and an element address. Under this address, they can be addressed by other NAUs.

The physical units are classified according to the type of format and protocols by means of which the PU is supported and according to the type of node which the PU contains. PUs of type PU.T1 are implemented in data station nodes, in which precisely one LU is situated (simple unit, data station). The PU.T2 characterizes an association node, in which several LUs are contained, for example corresponding to the number of connected I/O units (complex unit, association control unit). The NCP node, contains a PU.T4, and a PU.T5 is characterized by the implementation of an SSCP.

Logical units are distinguished into primary logical units (PLU), which are contained in the host nodes, and secondary logical units (SLU), which are implemented in peripheral nodes.

SSCP, host node PUs, and primary LUs are components of the TCAM/NF-MCP within a DFV system which operates with TCAM/NF.

PLUs are represented with TCAM/NF by the message handlers for units (DMH) and by a joint LU component. This MCP-internal LU component comprises LU services and other routines for session support. Although the TCAM/NF user programs can have logical addresses assigned to them, user programs do not represent LUs in the case of TCAM/NF.

Figure 3 gives a survey of TCAM/NF and the structure of the network.

3.2 Session Concept

The logical connection which is established and maintained between two NAUs according to certain rules is called a session. A session is uniquely determined by a pair of network addresses. There are three types of sessions for communication between different NAUs in one single area network:

- SSCP-PU session
- SSCP-LU session
- LU-LU session.

The sessions between SSCP and PU or LU are used to exchange control information. LU-LU sessions are used to exchange final user information. Before final users can be linked together, the corresponding SSCP-PU sessions and SSCP-LU sessions must be built up. These sessions are set up by the SNA commands Activate Physical Unit (ACTPU) and Activate Logical Unit (ACTLU). In the case of TCAM/NF, both commands are generated and outputted either automatically within the framework of initializing and activating the TCAM/NF MCP or as a consequence of a relevant TCAM/NF operator command from SSCP. An LU-LU session is initiated by TCAM/NF when certain TCAM/NF macro operands, specified by the user, cause automatic initialization, when a corresponding operator command was entered, or when the MCP recognizes that information is ready for output in a queue. However, the setting up of an LU-LU session can also be initiated from the SLU. In this case, this SLU transmits the command Selfinitialization (INITS) to the SSCP within the framework of the SSCP-PU. In both cases, the SSCP then sends the command Try Opening a Session (CINIT) to the desired PLU. On its part, the PLU builds up the command Link to Session (BIND) and sends it to the SLU. The BIND command specifies the protocols which must be adhered to with respect to setting up and maintaining the LU-LU session. To select the various protocols, there exists a so-called BIND image table, which represents a physical image of individual BIND commands that are characterized by a mode-name field. This table is made available in standard form through a TCAM/NF macro or it can be extracted by the user by means of TCAM/NF macros. When constructing the BIND command, the specification of the mode-name field is already required at the time of a request to the SSCP (TCAM/NF-side or in INITS). The SSCP thus searches the BIND image table and assigns this BIND image to the CINIT command. The

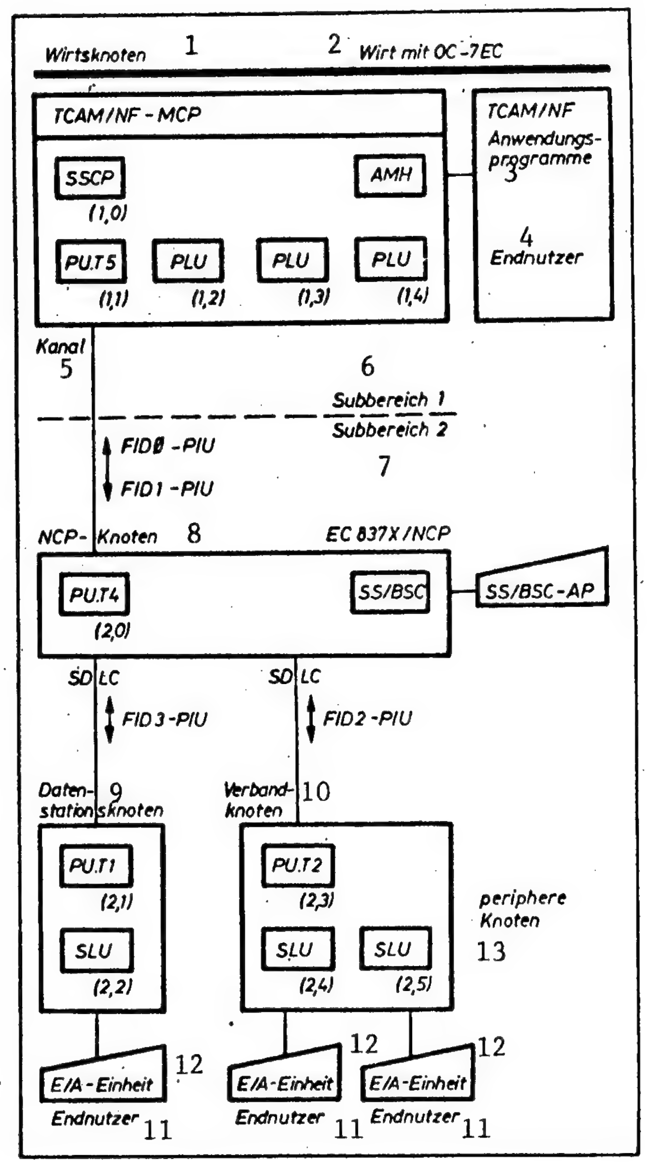


Figure 3. TCAM/NF and the structure of the network

1 host node
2 host with OC-7 EC
3 TCAM/NF user programs
4 final user
5 channel
6 subarea 1
7 subarea 2
8 NCP node
9 data station node
10 association node
11 final user
12 I/O unit
13 peripheral nodes

SLU can reject the BIND command, if it cannot adhere to the protocols. After a positive response from the SLU, final user data can be exchanged between the PLU which contains the primary session and the SLU which represents the secondary half session, in accord with the dynamically agreed upon protocols.

Besides these commands for opening a session, which are presented by way of example, there are other commands for controlling the session as well as for controlling the resources and the relevant requests during the sessions. It should be noted that the termination of sessions is also implemented by the host node and its NAUs.

The network architecture defines various types of LU-LU sessions which are essentially determined by the functional capabilities of the peripheral nodes. For example, it is possible for the PLU, in dependence on the LU-LU session type, to modify the received BIND command and send it back as an alternative BIND command. A presupposition for setting up an LU-LU session is in every case that both session partners specify the same LU-LU session type.

Further so-called transmission service profiles (TS profiles) and function management profiles (FM profiles) are defined in the BIND command. TS profiles define the commands allowable during the session and specify the possibility of sequential numeration for controlling every data transmission. FM profiles identify a predetermined set of special protocol directives.

3.3 Layer Structure

The information produced by the final users (e.g. user programs, data station operators, or information media within the input/output units) are also called the application layer. This information flows to other users of the network through a series of network nodes. In each of these nodes, certain functional layers are implemented. Each node contains the following six layers:

- NAU service management
- function management data services (FMD services)
- data flow control (DFC) NAU
- transmission control (TC)
- path control (PC)
- data linkage control (DLC)

The NAU service management for a single area network provides network services to control and manage the various session types and to coordinate the final user requirements regarding LU-LU sessions and their relevant interactions.

TCAM/NF uses three types of network services:

- Configuration services
  These activate, deactivate, and maintain the status of physical units as well as transmission lines, including those caused by restart as well as automatic network operation termination in overload situations. They are thus used for an updated determination and modification of the configuration in a network area. They are made available through SSCP and PU.

- Maintenance Services
  These provide aids for testing of nodes and transmission lines as well as for the collection and recording of error information within a network area, and are made available through SSCP and PU.
- Session Services
  These provide means for testing a logical unit or for requesting the operator to initialize or terminate LU-LU sessions.

The FMD services, in conjunction with the NAU service management, provide functions by means of which the SSCP can control and monitor the utilization and communication of network resources. The layer transmits FMD inquiries and responses to the individual NAU service management components. The special component representation services provide support for the user programmer and data station operator as well as for the formatting of data for screen display and printing.

The layer of data flow control makes available the protocols for the data flow between NAUs and takes care that they are adhered to. A distinction is made between

- full duplex (FDX)
- semiduplex flip-flop (with changeover indicator, HDX-FF)
- and semiduplex competition operation (HDX-CONT).

In addition, the use of brackets (bracket protocol) can be specified by means of DFC commands.

In this way, the chains of information units (RU chains, see 3.4) are collected together into larger units, so-called brackets. The use of brackets can be one-sided (monologue) or through both session partners (dialogue). The use of brackets in a session indicates that the monologue or dialogue within the brackets concerns a single subject; one of the two LUs is called the speaker and he has control over data traffic in the session. The other LU, called the requester, must request permission from the speaker before he can begin a bracket.

Through these protocols, the data flow control controls which of the semi-sessions can receive and/or send and it regulates the interaction of inquiries and responses. The transmission control implements synchronization and load regulation of data transmission on the level of sessions. For this purpose, it contains three components:

- Linkage manager (CPM)
  This provides equipment for communication of NAUs as well as session and network control components with their corresponding elements in the network. Included herein are the setting up, delivery, and evaluation of information-related control information, such as routing information, the performance of checks and sequence numbering of inquiries and responses, load regulation, as well as setting up exception status and sensing information, and the relevant reaction.

- Session control (SC)
  This is responsible for setting up and terminating sessions in the normal case and in the case of errors.
- Network control (NC)
  This regulates information exchange between joining linkage managers without formally setting up a session.

The path control is implemented once in each node. It is responsible for the transmission of information to the NAUs and determines the paths required for this purpose. The data linkage control controls the data transmission for one transmission line and takes care of error handling. In each node, DLC elements are implemented corresponding to the number of connected transmission lines.

The layers separate applications, devices, transmissions, and network control. Each layer provides variants of the services which it implements. These variants allow layers to be matched to various functions, situations, and products. It also allows the use of various combinations of variants for the most various sessions. A semisession represents this individual combination.

3.4 SNA - Information Units and Communication Between Layers and Nodes

The original message, as it was generated by the final user, is contained in the smallest SNA information unit, the inquiry/response unit (RU). The RU can contain either final user information (data) or system information. This means: instructions (inquiries) and responses to instructions. The name inquiry/response unit is derived from the function as a system message. The RU is transmitted from layer to layer of a node and, depending on its destination, from node to node. Here it is supplemented by prefixes in dependence on the direction of the information flow. These supplements concern layer-typical information. It is then evaluated and reduced in this respect, so that the final user receives only the message data and need not bother with the control information that is required for the information flow through the network. Four types of RUs can flow through the network:

- RU with function management data (FMD-RU)
- RU for data flow control (DFC-RU)
- RU for network control (NC-RU)
- RU for session control (SC-RU)

These RUs are used in layers of the same name or in their components, to implement the corresponding services. The actual final user data accordingly are contained in the FMD-RU. Corresponding to agreements in the BIND command, a function management head (FMH) can be carried in the FMD-RU. This function management head contains user-specific control information. Every FMD-RU, which is processed by TCAM/NF, corresponds to a TCAM/NF buffer. If a message is longer than a buffer, an RU chain is formed, which then represents the smallest information unit for restart after an error.

The layer-typical supplementations of the RU and the new information units generated thereby are shown in the following schema:

Final User	Message	
FMD services		
DFC		
TC (NC,SC)	RU	
TC (CPM)	RH + RU	= BIU
PC	TH + RH + RU	= PIU
DLC	LH + TH + RH + RU + LT	= BLU

BIU = basis information unit
PIU = path information unit
BLU = basis linkage unit

RH = RU head
TH = transmission head
LH = SDLC head
LT = SDLC appendix

Depending on the size of the BIU and the buffer size at the destination, the path control (PC) can segment the BIU. On the other hand, several PIUs can be contained in one BLU block through the path control.

In dependence on the destination of the PIU (channel linkage) or BLU (line connection), different PIUs are distinguished. These are characterized by the format indicator (FID). Here, TCAM/NF allows the following possibilities:

FID0 - PIU between PU.T5 and PU.T4 or respectively SS/BSC units
FID1 - PIU between PU.T5 and PU.T4 or respectively SNA resources
FID2 - between PU.T4 and PU.T2
FID3 - between PU.T4 and PU.T1.

4. Remarks regarding TCAM/NF-MCP

These remarks are intended for a programmer who is familiar with the access method TCAM. By means of Figure 4, they should provide an informative survey concerning the definition of resources made by TCAM/NF, and concerning a DFV system that is based on TCAM/NF. The following explanations will supplement the figure.

A large number of new functions of the MCP can be included in the MCP by coordinating various macro operands of the INTRO macro.

By means of OPEN, the opening of the following files is additionally possible:

- DCB for the 837X/NCP processor
- DCB for the 837X/NCP test point.

While the OPEN macro is being executed, the IPL for the 837X/NCP processor is initiated either automatically or according to operator command. Here, the NCP is loaded into the processor from the file SYS1.LINKLIB. At the READY time, the resources defined by the NCP are furthermore compared with those in the MCP. For this purpose, a resource specification table (RRT) must be generated during NCP generation. Likewise, this must be brought into the SYS1. LINKLIB. Only those network resources can be activated for which an agreement has been determined, for which similarity of names is a presupposition.

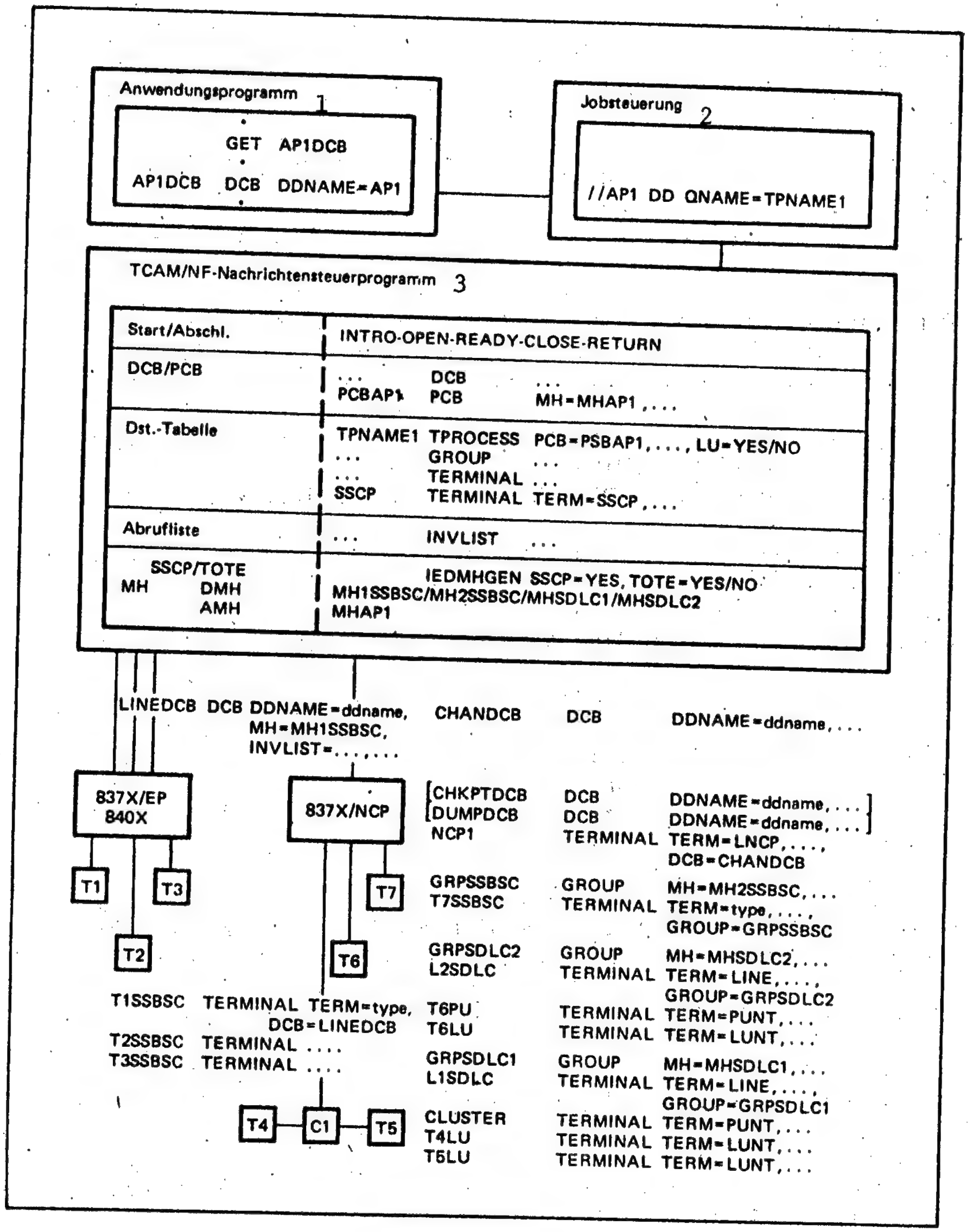


Figure 4. Survey of the TCAM/NF resource definition

1 user program
2 job control
3 TCAM/NF message control program

The INVLIST macro for defining the request list is generally not necessary for MCP resources since the corresponding functions are implemented by the NCP. Only under special conditions is it required for BSC data stations.

A number of new macros is made available for coding the message handler. Their function and application will not be discussed further within the context of these explanations. The DMH, in which FMD-RU and DFC-RU are processed, are especially important here.

Within the framework of the SSCP-MH, so-called system services for non-formatted inquiries (USS) and system services for formatted error indicators (FSS) can be utilized. By means of these services and the associated USS and FSS conversion tables, non-formatted entries can be effected through the SLUs. These are then converted into formatted SNA inquiries and are transmitted through the SSCP. Alternatively, formatted error indicators (SNA responses) are converted into non-formatted error messages, and are outputted through the SLU. In conjunction with the standardized SSCP-MH, TCAM/NF makes available standardized conversion tables. Here, one can, for example, input the hexadecimal, occupied, formatted request, up to about 20 bytes long, for the SNA command Selfinitialization and Selftermination, by inputting in the non-formatted form

- INITS luname, modename or
  LOGON lunmane, modename and
- TERMS luname

Furthermore, macros are made available by means of which the conversion tables can be generated by the user.

The function of the modename field was discussed in Section 3.2. In the luname field, the address of the desired session partner must be specified. In the case of TCAM/NF, this can be, for example, the name of a user program, a destination queue, or a DMH message handler, or the name of the actual message partner.

5. Final Considerations

This paper has attempted, by means of selected explanations, and under perspectives relating to applications and access methods, to provide the insight into the new application possibilities and functions of the DFV access method TCAM/NF.

It should be especially emphasized that TCAM/NF represents the only DFV access method of the ESER which supports the system and network resources of the DFV. The resultant advantages were discussed in Section 2.

The centerpiece for opening up the new application possibilities is the network architecture of the ESER and its implementation in terms of devices and programs, which must always be regarded as a unit.

With the network architecture of the ESER, perspectives are indicated which extend from the single area network of simple hierarchical structure with local DFV processors through extended hierarchical structures and remote DFV processors, as far as multi-area networks (computer networks, with inter-linked structures under the central control of ESER data processing systems). Along this path, the availability of the DFV access method TCAM/NF represents a first step, as an integral component of the operating system complex OC-7 EC. Basic principles pertaining to access methods are contained within it, to control a DFV system according to the network architecture of the ESER in its current and changing state.

The ESER network architecture provides the concept for a complete DFV system. By the use of means furnished for this purpose, the user is greatly facilitated in planning and implementing a comprehensive DFV operation. In contrast to frequent previous practice, individual DFV components need not be distinguished here with, for example, special TCAM/NF-MCP. Rather, in analogy to the OS/ES system programmer, the TCAM/NF system programmer must plan and implement the complex possibilities and requirements of a computer center and its users. Thus, the actual DFV users can concentrate fully on their specific tasks when using the given interface relations and operating directives. For this purpose, the ESER network architecture guarantees broad flexibility. From the point of view of hardware, the network architecture of the ESER generally on the one hand presupposes the intelligence of data stations (association and data station nodes) and, on the other hand, the opportunity is offered to the user for their comprehensive application-related use.

With the displacement of the line control into the DFV processor, the EDP system is relieved of subordinate tasks, and CPU time is gained for complex control (control central SSCP and subordinate services) and maintenance (function TOTE of TCAM/NF) of the DFV system.

Considered altogether, the implementations of ESER network architecture, as they apply to programming and especially to hardware and transmission, provide an increased data throughput with reduced response time behavior, greater system reliability, and more services for the user.

POC Multiple-Computer Control System

East Berlin EDV ASPEKTE in German No 3, 1984 (signed to press 30 May 1984) pp 40-44

[Article by Wolfgang Hoesel, Magdeburg VEB Data Processing Center]

[Text] The constant increase of computer power by more intense utilization of resources plays a major role for practical computer operation in computer centers.

International experience makes clear that a spatially concentrated use of several data processing systems (EDP systems) can achieve a significant

increase of job throughput if the individual computers are coupled and if they operate under a multi-computer control system that supplements the OS/ES operating system.

This article describes the basic mode of operation of a multi-computer system under the control of the multi-computer control system POC, which is furnished as a component of Type 2 for the OS/ES.

The functional scope presented below is implemented in Edition 3.1 of the POC.

POC 3.1 is developed for collaboration with the operating systems OC-7 EC and OC-6.1 EC Modification 9. Delivery of Edition 2.2 of the POC for the operating system OC-6.1 EC Modification 9 is planned for the year 1984. In this edition of the POC, TSO connections and remote job processing are not supported.

1. Configuration and Basic Mode of Operation of the Multi-Computer System

A multi-computer complex controlled by POC consists of a lead computer as well as one or more working computers. The individual EDP systems are coupled by way of hardware through a channel-channel adapter (KKA).

Such a computer linkage is generally called a local multi-computer system.

Furthermore, selected external memory units can be connected with the coupled EDP systems through multichannel device control units. The local multi-computer system itself can be asymmetric, which means it can consist of different central units of the ESER, all of which, however, must work with the OS/ES operating system. The task of the lead computer is to control and to monitor the entire system centrally. The lead computer distributes the work to the working computer from a central job chain file. All disk and tape memories that are provided as data media are managed by the lead computer. It automatically assigns these devices to the jobs and thus to the corresponding working computers. The entire system input and system output runs through the lead computer.

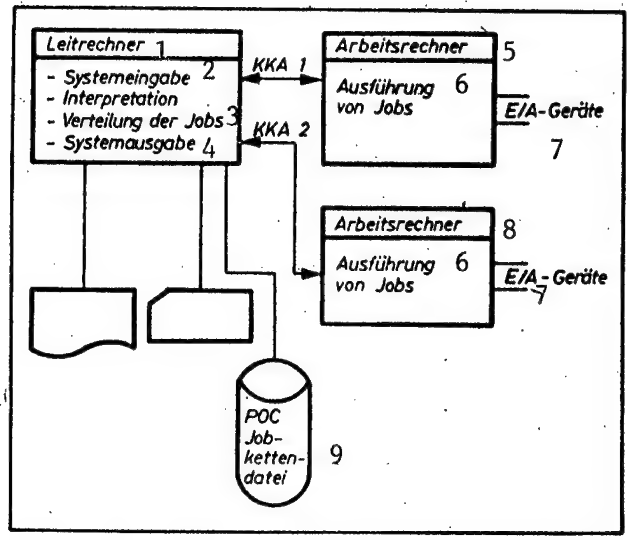
The user programs are executed on the working computers.

Figure 1 shows the basic mode of operation of a multicomputer system under POC control. The external devices, which means all tapes, disks, printers, card readers, and punchers are always collected together into device groups.

The device group which takes care of system input and output (card reader, card punch, and printer) is installed at the lead computer. Furthermore, disks for the operating system and the POC job chain file as well as tape memories for processing dynamical auxiliary programs are required at the lead computer.

Tape and disk groups are allocated to each working computer. The use of selected devices by several working computers is possible here, e.g. SHARED DASD. Each device group can be provided with an operating unit. Messages are outputted there which concern the device group. The operating units must

Figure 1. Mode of operation of a multi-computer system



1 lead computer
2 system input
3 distribution of jobs
4 system output
5 working computer
6 execution of jobs
7 I/O units
8 working computer
9 POC job chain file

be connected to the lead computer. An extensive operating unit service is contained in the POC.

An operating unit of the lead computer is the main operating unit of the multi-computer system. From there, the entire system is controlled. The lead computer function can be implemented both alone and also jointly with a so-called local working computer section on an EDP system. This mode of procedure is used when the computer association consists exclusively of powerful EDP systems. For the behavior of the computer association in case of a catastrophe, this mode of operation is very significant. In local operation, one works with a fictitious KKA. With the lead computer-working computer principle, the control system POC creates the conditions which allow operation of the multi-computer system like a single computer system:

- A job chain file exists, which is managed by the lead computer, and which implements system input/output through the lead computer.
- All disk and tape memories which are connected to the working computers are centrally managed by the lead computer.
- There exists a central interface for operating the computer complex.

2. Structure and Availability of the Control System

POC is a system, based on the OS/ES, for multi-computer support. It runs as an endless one-step-job of higher priority under the control of the OS/ES on the lead computer. The basic operating system is an MVT or SVS of the OS/ES in the configuration OC-6.1 EC Modification 9 or BPS of the SVS OC-7 EC. The use of various control program configurations (MVT, SVS) within the multi-computer system is possible.

The component POC is marketed as a component of Type 2 on a separate distribution medium. The operating system OS/ES, which is used as the basic system, must be ready for work with POC. In particular, KKA and fictitious magnetic tapes must be defined for this system.

The requirements for operation with POC are formulated through the following generation macros:

IODEVICE
This macro must be specified for each real and pseudo KKA as well as for the fictitious tape memories, which are required for the allocation of input stream files and system output files for a running job.

UNITNAME
The fictitious tape memories must be collected together into a device group. CTC is necessary as the device group name.

SCHEDULR
As an alternative operating unit, a real or pseudo KKA must be specified. To connect the multi-computer system, one must specify OPTIONS=POC.

SECONSLE
With multi-computer operating unit support, a SECONSLE macro is necessary for the KKA as alternative operating unit.

For the multi-computer control system POC, all parameters which depend on the device configuration and which influence the control possibilities are specified in the initialization control statements.

Thus the POC system is very flexible. It can be adapted with each initialization to a possibly changed environment.

The multi-computer system consists of a resident part (POC core) and a series of dynamic auxiliary programs. The latter are loaded as they are needed. Through the POC core, the following functions are implemented among others:

- access to the POC job chain files
- control of the operating unit
- processing control and call of dynamic auxiliary programs
- allocation and release of main storage space on the lead computer
- interrupt handling for the KKA.

For the individual functions of POC, there are auxiliary programs which are loaded dynamically.

If sufficient space is available in the main computer, the most frequently needed programs can be made resident. The work of the system is controlled through tables. These tables contain specifications concerning each connected working computer, the processing status of each job, the occupation of external devices, the availability of external devices, etc. They are stored in the POC job chain file and are constantly updated. In this way, the current status of the POC system can be reestablished at any time. The management of the POC job chain files, and, in conjunction therewith, also the management of the buffer complex, is effected by special routines. These control the data flow between the lead computer and the POC job chain files. The storage space for a file in the POC job chain file is distributed by a special algorithm in such a fashion that the arm motion during writing is minimal. The data records being written are blocked into records in a buffer complex. These records are best suited for the memory units being used (EC 5061, EC 5066, EC 5067). An efficient transmission rate is thus achieved.

Furthermore, a series of dynamic auxiliary programs for frequently recurring functions are also part of the system. Such dynamic auxiliary programs implement, for example, the following applications:

- transmission of punched cards to the punch, to the printer, or on a tape
- transmission of a tape file to a card punch or to the printer
- the doubling of tape files
- the initialization of tapes.

An arbitrary number of auxiliary programs can be processed in parallel on the lead computer, as soon as the main memory is free and the necessary hardware is available. The programs can be called either by the operator or from the job.

The read-in jobs can contain not only job control statements but also POC control statements. By means of these control statements, the user can, among other things, select a special working computer to execute the job, can use the special capabilities of the POC printing service, or can call special dynamic auxiliary programs.

3. Job Processing in the Multi-Computer System

The jobs which are transferred to the multi-computer system POC are subdivided into processing segments. Here, each segment is processed by a different lead computer service. The following segments are distinguished:

Preprocessing segment
Working computer segment
Post processing segment

The actual job processing is implemented on the connected working computers or on a local working computer under the control of the OS/ES. The lead computer takes care of the tasks of job preprocessing and job postprocessing. By means of this preparation, it is possible that the individual segments run asynchronously and the individual segments of various jobs are executed in overlapping fashion (Figure 2). The POC divides the jobs either automatically or corresponding to the statements of the programmer. By means of the POC control statement PROCESS, the program can specify the division and sequence of the segments. A job in which the PROCESS control statements are missing is regarded as a standard job. Standard job means that the job will be processed in a standard sequence of job segments. The standard sequence consists of the job segments

- input segment — preprocessing segment
- working computer segment
- print segment
- punch segment — postprocessing segments
- release segment

The processing of a standard job under POC control is shown in Figure 3. Starting from reading the job input stream, the following steps are traversed:

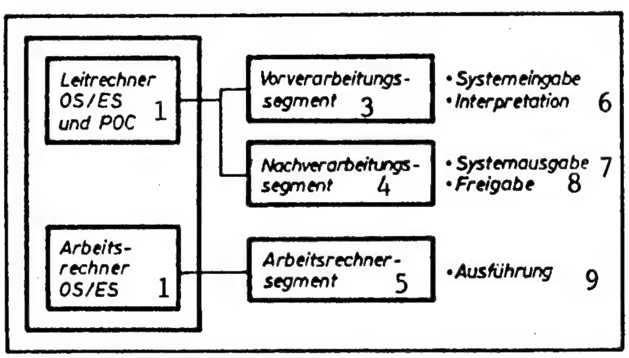


Figure 2. Job processing under POC control

1 lead computer OS/ES and POC
2 working computer OS/ES
3 preprocessing segment
4 postprocessing segment
5 working computer segment
6 system input
7 system output
8 release
9 execution

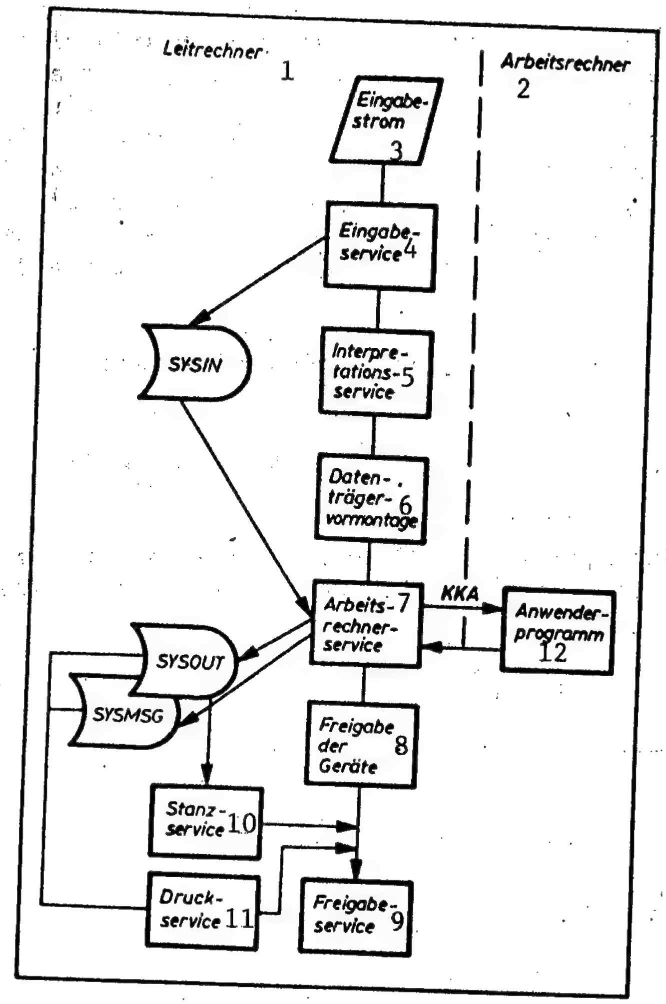


Figure 3. Processing of an OS/ES standard job

1 lead computer
2 working computer
3 input stream
4 input service
5 interpretation service
6 data medium premounting
7 working computer service
8 release of the devices
9 release service
10 punching service
11 printing service
12 user program

Input service

POC offers the possibilities of reading a job input stream from tape, disk, cards, or floppy disks. The read-in job streams are stored in the POC job chain file. Job and input stream files are stored separately.

Interpretation service

The interpretation service interprets the jobs contained in the POC job chain file and determines the resources required for processing them on the working computer as well as the dynamic auxiliary programs that must be executed on the lead computer. The OS/ES interpretation program is loaded during the initialization of POC as a subtask. It is possible to process several interpretation programs simultaneously. In the interpretation of jobs, system output files are assigned to the channel-channel adapter. Procedural references which occur are resolved from the procedure library of the lead computer (Figure 2).

Working computer service

The device dispatcher for the working computer is a section of the working computer service. It is active before and after execution of a job on a working computer and it takes over control of the assignment and release of devices as well as the mounting of data media for devices which are required to execute the job on a working computer. Once the mounting has been accomplished, POC tests whether the mounting was performed correctly. In the case of a defect-free mounting, the units are definitively assigned to the job, and the job is characterized as ready for processing. Through the premounting of the data media, every job is prepared for execution in such a fashion that it no longer needs to wait for the data medium before the actual execution and thus will not block computer capacity (Figure 3).

After the data media have been mounted, the job is planned for processing on the working computer. Here, the planning algorithms that are specified in the initialization of POC are taken into account. Among other things, it is specified how much computer-intensive, how much I/O-intensive, and how many balanced jobs represent an ideal load for the working computer. The working computer service tries to achieve this job mix as frequently as possible. If a job has been selected, the working computer service assigns execution resources to this job (initiator, main memory), and transmits the OS/ES control blocks into the OS/ES job chain file of the working computer via the KKA.

During execution, the SYSIN file of the job is read by the POC job chain file. For the resulting system output, a SYSOUT file is generated on the POC job chain file. During job execution, the working computer service takes over the role of an operator of the working computer. After job execution has been completed, the working computer service stores the system message block that arose during the execution in the POC job chain file.

After the job has been processed, all devices occupied by the job are released.

Printer service

After a job has been completed, the printer service takes over the output of system messages and system output files at the lead computer. The printer service permits working with different printing forms, print control strips, and print chains, which are specified by the programmer. The printer is selected on the basis of the required printing form. In this way, operator labor is considerably reduced. In contrast to the OS/ES, the system files of a job can be printed on various printers. The records being printed out can have a fixed, fixed block, variable, variable block, and undefined record format.

The printers that are supported by the printing service must all be connected to the lead computer. If an error occurs during printing, the operator can have the output printed by another printer with appropriate equipment.

Release service

The release service is performed as the last segment of each job. In this phase, the job specific invoicing information is outputted on cards or in the POC job chain file. By means of a user routine, the invoicing information can be outputted in private files. After the release service has been processed, no information concerning the job is available any longer.

Besides the standard job processing described above, there is a processing sequence for non-standard jobs.

A non-standard job is characterized by a segment sequence which differs from that of the standard job (i.e. omission of working computer segments, simultaneous output of a file, calling dynamic auxiliary programs within the job). Individual segments are planned in the sequence prescribed by the PROCESS control statements. The processing segments that are called in this fashion are processed exclusively on the lead computer. The linkage to a working computer is made by a non-standard job if an EXEC statement is present in the job stream.

4. Additional Functions

Besides the actual job processing, POC also contains functions which satisfy the system requirements that are imposed by a computer center.

Job network control

The principles of job network control are used when several jobs must be processed in a certain sequence. All mutually dependent jobs are assigned to a group, the job network.

The assignment of a job to a particular job network is effected by including the POC control statement NET in the respective job. The main function of

the job network control consists in releasing a job for job planning after the successful processing of one or more preceding jobs. The job network control is started automatically when a POC control statement NET is recognized. The operator can trace or change the status of a job in a network by means of a command. The job network control contained in POC is compatible with the job network control of the OC-7 EC.

Deadline Planning

Deadline planning is a method for planning jobs whose processing must be begun by a certain time. Through the POC control statement MAIN, it is possible to specify a type of deadline planning for this job. If the input service recognizes the parameter DEADLINE in a MAIN control statement, it automatically calls the dynamic auxiliary program DEADLINE, if it is not already active. The job itself is inserted in a chain of deadline plans. The program DEADLINE increases the priority of the jobs in this chain in the intervals that are provided by the selected types.

TSO support

By means of special POC interface routines, the TSO user is offered the capability of storing the jobs that have been submitted with the SUBMIT command in the POC job chain file. During execution of a job on a working computer, the TSO user can interrogate the status of the job and/or can terminate the job.

Internal job processing

Internal job processing can be used, for example, to transmit job streams from private files of the user into the POC job chain files. The transmission is made through the KKA. The transmitted data (e.g. job streams) are in this case furnished to the POC input service. The jobs transmitted from the TSO to the POC, for processing in batch operation, are likewise transmitted into the POC job chain file by using this capability.

The generating job, which is always running on the working computer, transmits by means of a special program a file with corresponding data via the channel linkage to the lead computer. The lead computer makes available the transmitted file, as needed, to the input service or to the printer service.

Making available additional dynamic auxiliary programs

For special requirements, the user himself can write dynamic auxiliary programs and can insert them into the POC. These programs are written in the assembler language by using special POC macros and with consideration of the rules that are valid for this programming.

Possibility of remote processing of jobs

POC supports the possibility of remote input of jobs for processing in multi-computer systems (e.g. from an FRTS machine of the SVM/ES). Here, jobs are subject to the same programming conventions as those which are brought into the system locally. System output can optionally be transmitted between the remote stations as specified by the sender of the job, or can be processed locally.

5. Restart Functions

The multi-computer system is restarted by an appropriate initialization.

An essential point for a restart is that all tables in the POC job chain file are stored on disk and are always maintained at the current status. If, during the restart, it is determined that unfinished jobs or dynamic auxiliary programs are still in the system, a restart is also implemented for these. If the job was in a working computer segment, the type of restart is specified by initialization control statements as standard for all jobs or is specified by means of POC control statements in the job. By means of these control statements, it is possible to request termination of a job, placing the job into a waiting state, or an automatic job restart. If no automatic job restart is specified, all already existing system output files can be printed or punched. Besides restarting the multicomputer system, it is possible to restart the dynamic auxiliary programs from the printing service while the multi-computer system is operating.

6. Advantages of POC Application

In comparison to the separate operation of an EDP system, the use of the EDP system as a multi-computer system through control of the component POC yields the following advantages:

- The work division in job processing between the lead computer and the working computer achieves a better workload for the resources, because the working computer is relieved from the slow functions of job input and of output of system files.
- By organizing and managing a single common job chain file for all EDP systems, one achieves an optimal job and resource plan.
- POC offers the capability of forming device groups from external devices, which can be provided with their own operating units. If this capability is utilized and if the device groups are installed physically in the vicinity of the archive or paper storage depot, the work flow in the computer center can be significantly improved.

Program Development with PTS

East Berlin EDV ASPEKTE in German No 3, 1984 (signed to press 30 May 1984) pp 52-55

[Article by Bernd Lilpopp, Klaus Wagner, Berlin VEB Center for Applications Research]

[Text] The main application of the component PTS of the system of virtual machines (SVM) consists in the development of programs which are intended to be processed under the control of a batch processing system of the OS/ES.

This article gives a survey concerning the capability which PTS offers in this connection.

Starting from the steps of program development, those functions are mainly explained which were implemented new in the PTS of Edition 3.0 of the SVM as compared to Edition 1.2.

1. Introduction

The system of virtual machines SVM is available as a component of the OS/ES Edition 7, for models of the ESER Series 2. By constructing and managing the virtual machines, this operating system guarantees the parallel processing of various users on one EDP system. The concept of virtual machines was described in /1/. The article in /2/ gives a survey of Edition 3.0 of the SVM.

An important application of the concept of virtual machines is subscriber operation, which means the processing of independent jobs in the dialogue mode. Here, the control program (CP) takes over the organization of the parallel operation, while the dialogue-capable programming and test system (PTS) serves as the controlling system in the participating virtual machines and guarantees the dialogue work of the particular user.

The application possibilities of the CP/PTS subscriber operation are manifold. Thus, it is possible, under the control of the PTS, to process programs and smaller programming systems, also including dialogue oriented programs. Furthermore, the construction and maintenance of documentation can be implemented by means of the text processing system TEPROS.

The main application of PTS consists in the development of programs which are intended to be processed in batch operation under the control of the BPS or SVS. This application is the main subject of the present article.

The component PTS of the SVM, Edition 3.0, fully guarantees upwards compatability to Edition 1.2, and it implements copious functional extensions. These are used mainly for the further support of program development.

These changes and extensions essentially concern the following points:

1. Expansion of the file system. For the user, this means that instead of having, as before, 10 virtual disks (A-G, S, Y, Z) he now has 26 virtual disks (A-Z), and that the size restriction for PTS files has been expanded from 65535 records to 2**31-1 records and thus has practically been obviated.

2. Expansion of data communication with the BPS and SVS, and the simulation of functions from these systems. Sections 3 and 4 will indicate these possibilities.

3. Development of the expanded editor XEDIT. In comparison to the existing editor, XEDIT has available a significantly greater functional scope. The main features of XEDIT are described in Section 2.

4. Development of the expanded procedure interpretor EXECE. In comparison to the existing EXEC processor, the EXECE processor has a significantly larger functional scope. As regards the main features of EXECE, see Section 5.

5. Implementation of a HELP function. By means of the PTS command HELP, the user can have displayed information concerning the format, the operand, and the application directives of the CP commands, the PTS commands, the EDIT, XEDIT, and DEBUG subcommands, selected control statements of the processors EXEC and EXECE, as well as selected SVM messages at the virtual operating unit.

6. Implementation of a program push-down memory. This program push-down memory is separate from the push-down memory of the operating unit. As a result, processing reliability of procedures is increased. The possibility of defining levels in the program push-down memory supports the structurization of user programs.

7. Detail expansions in existing commands. Thus, for example, in the command COMPARE, equal signs can be specified in place of parts of the second file designation, and the scope of output of unequal data records can be limited. In the commands LOAD and START, support of the matrix module was included. The conception and implementation of the above expansions was primarily based on experience which was gained in the previous use of the SVM/ES, Edition 1. Because of upwards compatibility, no conversion of existing programs and procedures is necessary on the part of the user. For new developments, the application of the new and more efficient functions, for example the use of EXECE instead of EXEC, is recommended.

2. Structure and Change of Source Texts and Test Data

In the process of program development, it is necessary to construct and change source texts and test data. These data are furnished as PTS files and are processed with the commands of the PTS file system. The most important of these commands are:

READCARD - read a file from the virtual card reader
PRINT - output a file to the virtual printer
PUNCH - output a file to the virtual card punch
COPYFILE - copy a file between PTS disks
RENAME - rename files
ERASE - delete files
COMPARE - compare files
TAPE - unload and again store files on tape

MACLIB - construct and manage macro libraries
TXTLIB - construct and manage object program libraries

If existing source text and test data on the data media of BPS or SVS are to be used, the following commands are also important:

MOVEFILE - transport files between virtual units, e.g. from OS/ES disks to PTS disks
TAPEMAC - construct a PTS macro library from a PDS file that has been rolled out onto tape
TAPPDS - construct PTS files from inventories of a PDS file that has been rolled out onto tape
LABELDEF - specify tape identification records.

The most important function of the PTS files system is the editor function. The basic editor EDIT is implemented in the PTS of Edition 1 of the SVM. It is called with the PTS command EDIT and, by a large number of subcommands - freely expandable through the EDIT procedures - it guarantees the construction and preparation of PTS files. The expanded editor XEDIT was implemented on the PTS of Edition 3.0 of the SVM. This is called with the PTS command of the same name. The functional scope of EDIT is a subset of the functional scope of XEDIT. The editor EDIT was retained in the PTS, so as to facilitate the user becoming familiar with Edition 3.0, and so as to avoid changing existing procedures which contain EDIT calls. The new quality of file handling which is achieved with the expanded editor XEDIT is characterized by the following features:

1. Utilization of full video screen support. The entire video screen is managed by the XEDIT supervisor. Thus, it is possible to make changes not only in the current line but on all points of the display area, by means of the cursor. After the entry key has been activated, all changes entered on the screen become effective in the main storage copy of the file. Because positioning operations are now deleted, this perceptibly reduces the processing times as well as the number of line activities for remotely connected data stations.

2. Implementation of functions of pattern recognition. In the editor EDIT - subcommands LOCATE and FIND - it is only possible to search according to fixed context patterns. With the expanded editor XEDIT, the possibilities for constructing patterns were expanded primarily in the directions of

- combinations of patterns
- specification of search direction
- specification of a symbol for "arbitrary character"
- ignoring the number of blanks in search patterns
- defining patterns which are situated in several lines.

Furthermore, in many subcommands it is possible to specify the action range through a target line that is specified through a context pattern.

3. Possibility of simultaneously processing several files. These files are managed in a file ring. The transition between the files takes place without leaving the XEDIT control level.

4. Possibility of subdividing the physical screen into several logical screens. Various files of the file ring, as well as various parts of one file, can be displayed and processed on these logical screens. As applications of this function under the perspective of program development, one could conceive of the following in connection with two logical screens:

- the parallel display of a LISTING file (error messages) and the source text
- the parallel display of constant definitions and executable statements of a source program.

5. Use of XEDIT macros. Sequences of XEDIT subcommands, PTS commands, and EXECE control statements can be collected together to form files of the type XEDIT, and can be activated by entering their names and possible parameters at the XEDIT control level. The interpretation of these macros is made through the EXECE processor described in Section 5. While communication between EDIT and EXEC is limited to the push-down memory of the operating unit, a direct interface is implemented between XEDIT and EXECE. From this, and from the larger functional scope of EXECE, advantage is derived for the new processing macros.

6. Capabilities of the UPDATE function. If the selection condition UPDATE is active, the changes made during the XEDIT session are not physically made in the file itself, but are collected in a file which, processed by the PTS command UPDATE, yields the changed file content. This makes it possible for the user to use both the advantages of interactive work and to obtain the changes in a form which is advantageous for documenting the current development stage.

7. Implementation of a large number of individual functions. As an example one need only mention here the subcommand SPLIT/JOIN. This is used to subdivide the current line into several lines or to chain follower lines to the current line.

The XEDIT language comprises 138 subcommands and can be freely expanded by macros.

3. Translation and Execution of Programs

3.1 Translation

The following compiler and programming systems of the OS/ES are available under the control of the PTS:

- ASSEMBLER 2
- programming systems PL/1-test compiler (PL1TC) and PL/1 optimization compiler (PL1OC)
- programming systems FORTRAN with the compilers FORTRAN CC (FORTCC) and FORTRAN SE (FORTSE) as well as the optimization compiler FORTRAN OE

(FORTOE) and a test compiler
- the programming system COBOL
- the programming system PASCAL.

It should be noted here that these programming systems are distributed separately. The runnability of the programming system in PTS is established by interface programs which are a component of the distribution tapes of the corresponding programming systems. In particular, the necessary compiler files are here defined by the generated commands FILEDEF and the compiler call is supported in the form of a PTS command. By creating the appropriate interface programs, further compilers of the OS/ES (ALGOL, RPG) and also compilers for user-specific special languages or other processing programs can be connected to PTS. As regards the type of compiler utilization for translations and for testing programs, one can distinguish between batch compilers and test compilers. The work with batch compilers (Assembler 2, PL1OC, FORTSE, etc.) is characterized as follows: After the errors have been detected, the source text and/or the test data are changed with a PTS editor, and the process of translation and/or execution must be initiated anew. The advantages of PTS dialogue here appear "only" in the points:

- detection and elimination of several errors within one session at the data station
- dynamical test strategy by entering test data through the data station
- the saving of printing paper.

The test compiler such as PL1TC, on the other hand, are characterized by collecting together into one integral process the stages of translation and interpretive processing. During the course of this process, the user can intervene interactively by means of subcommands.

As a result of program translation with a batch compiler, one obtains a PTS file of the type TEXT which contains the object module.

3.2 Execution

Object modules can be brought to execution by means of the PTS commands LOAD and START. During these executions, as already explained in /1/, most functions of the BPS and SVS are simulated by the PTS. This concerns both supervisory functions, for example LINK, and also functions of data management. Restrictions of this simulation result from the concept of virtual machines (remote data processing) or the concept of the PTS as a single user system (subtask operation). Programs which use such functions can only be translated under PTS and can be executed under BPS or SVS.

As in the PTS of Edition 1.2 of the SVM, so also in Edition 3.0, the access methods BSAM, QSAM, BDAM, and BPAM are supported. Files of the corresponding organizational forms can be read, constructed, and updated by programs that are running under the control of PTS, if they are managed as OS/ES simulated PTS files (file mode digit 4). For sequential and subdivided files, it is also possible to read from virtual disks of the BPS and SVS (below designated briefly as OS/ES disks).

In the PTS of Edition 3.0 of the SVM, a support for a generalized access method for direct access memory (VSAM) and the associated utility program (AMS) were also implemented.

The user has the capability of reading and constructing VSAM files, where complete data compatability with SVS is guaranteed.

VSAM files and VSAM catalogues are defined in PTS with the command VSAMDEF.

The functions of the VSAM utility program are made available by the PTS command AMSERV.

It should be noted that the functions of the memory space management of the SVS for direct access memories are not simulated in the PTS. From this it follows that the setting up, deletion, and expansion of VSAM data areas must be effected under the control of an SVS system. Furthermore, the VSAM main catalogue is handled in PTS only as a private VSAM catalogue. As a consequence, the functions of the VSAM utility program, which require a modification of the main catalogue, cannot be executed in PTS (e.g. EXPORT with the option DISCONNECT).

However, these restrictions are not decisive, since they do not significantly influence the objective of VSAM support in PTS, namely the support of program development.

3.3 Execution of Load Modules

Up to now it has not been possible to execute load modules under the control of the PTS, if these modules were constructed under the control of a BPS or SVS.

Two commands are made available for this in Edition 3.0.

By means of the command MOVELOAD, the user can bring load modules into PTS files of the type TEXT. These files are brought to execution by means of the commands LOAD and START.

By means of the command OSRUN, direct execution of the load modules from OS/ES disks is possible. Parameters can be transferred here also.

Furthermore, it is also possible to load and execute load modules through the macros LOAD, LINK, and XCTL.

In the PTS of Edition 3.0 of the SVM, expansions of the command DEBUG were implemented. Hold points are not automatically eliminated after they have been reached. In the subcommand BREAK, the user can specify how often the hold point can be reached before it is eliminated. Furthermore, subcommands (e.g. DUMP) can be specified which are to be executed automatically when a hold point is reached. Hold points can be explicitly eliminated by means of the subcommand DELETE.

The above-mentioned DEBUG extensions particularly facilitate the testing of program loops.

4. Data Communication with the BPS and SVS

The main application of the PTS consists in the development of programs, which are intended for processing under the control of a BPS or SVS. For this reason, data communication between the PTS and these systems is especially important and forms a focal point in the further development of Edition 1.2 of the SVM.

A method of communication consists in the utilization of CP functions, especially file buffering. In practice, user solutions have established themselves in which job streams are built up under PTS control and are transmitted to a virtual machine for batch processing, and where the processing results are then returned. On the other hand, the capabilities of the CP functions VMCF are still used too little.

To adapt to the tape processing in the BPS and SVS, the processing of tape identification records was included in the BPS of Edition 3.0.

It is possible

- to construct and process standard identification records
- to process standard user identification records and
- to process non-standard identification records.

The tape identification record processing is supported by the PTS commands TAPE, TAPEMAC, TAPPDS, the PTS macro TAPESL and by the simulation of the macros OPEN and CLOSE during the execution of OS/ES programs. The content of the identification records are specified by the new PTS command LABELDEF. The special type of identification record processing is fixed by the new options of the PTS command FILEDEF during the definition of the tape files. The support of processing tape identification records is not only important for data communication with the BPS and SVS. Even with a "pure" PTS application, these functions can be used to increase the reliability of computer operations by appropriate organizational and technological measures. When processing files on OS/ES disks, up to now it was only possible to read files and to construct them as PTS files with the command MOVEFILE. In the PTS of Edition 3.0, these possibilities were expanded by the commands MOVELOAD and OSRUN.

Furthermore, in the PTS of Edition 3.0, it is possible to write files on OS/ES disks. One possibility was already presented when describing the support of the access method VSAM in PTS.

Furthermore, two new commands were developed for this purpose. The command MOVETSET writes PTS files on tape in a format which is suited for re-storing an unloaded file through the utility program IEHMOVE. The files can be written on tape without identification records or with standard identification records. The command MOVEDSET transfers PTS files directly on an OS/ES disk.

The following can be transferred:

- ordinary PTS files
- simulated PTS files (with MOVEDSET also DA files which are brought over into sequential files
- the libraries MACLIB or TXTLIB
- inventories of the libraries MACLIB, TXTLIB

The input and output files are here defined by the commands FILEDEF. During the transmission, blocking can be changed, but not the other characteristics of the file organization.

The two above commands support the user during the further processing of programs and data prepared in the PTS under the control of a batch processing system.

5. Procedure Processing

One of the most important functions of the PTS is procedure processing, by means of which all steps of program development are also supported.

In the PTS of Edition 1.2 of the SVM, the EXEC processor is available for procedure processing. Practical experience has shown that users use procedures to an increasing extent and thus also implement complex calculational processes, for example test technologies.

When working with the EXEC processor, the limits of the application possibilities have been recognized. This was the main motive for developing an expanded EXEC processor EXECE. The most important characteristics of EXECE will be briefly presented below:

1. The processing of character chains of variable link up to 255 characters. EXEC is limited to 8 characters, and the elimination of this restriction is important for many applications, for example for the construction of job streams.

2. Two-way communication between the procedure and the program. In EXEC procedures, programs can be called and parameters can be transferred to them. In EXECE, it is additionally possible to identify parameters as variable during the call. The called program can change these parameters and their new values are subsequently available to the calling procedure.

3. Subprocedures. It is possible to form one or more subprocedures within one EXECE procedure. These subprocedures can be called as a function or as a procedure. No variable environment of its own is formed for a subprocedure. For this reason, the user of all variables of the main procedure is possible.

4. Improved processing efficiency. In the existing processor, the file is read record-by-record in the sequence of interpretation. With copious procedures and cyclic processing, this method is relatively inefficient. EXECE uses a line memory in which the lines are stored which are to be interpreted repeatedly with great probability, for example, lines between the $FOR and $END. Storage takes place in an intermediate code whioh can be interpreted faster than the source code.

5. Statements of structured programming. The most important elements of structured programming

&IF...&THEN...&ELSE
&DO...&UNTIL
&FOR...&TO...&BY

exist as EXECE control statements.

6. Reading and writing from/into PTS files. Direct access to PTS files is supported. Thus, for example, through the statement

&READ FILE BSP DATA A 8 ARGS

The eighth record of the file BSP DATA A is read in and its words (separated by blanks) are assigned to the reserved variables ?1, ?2, etc. as values.

The EXECE language is very compact and is freely expandable through the two-way communication and through the subprocedure concept.

The control languages of EXEC and EXECE are not compatible. But it is possible to call an EXEC procedure within and EXECE procedure and vice versa.

References

/1/ Lampenscherf, S.; Schroeder, A.; Wagner, K.: System of virtual machines (SVM/ES). rechentechnik/datenverarbeitung (Computer Technology/Data Processing) 18 (1981) 2, p. 23.

/2/ Heinecke, K.; Schroeder, A.: System of virtual machines - SVM, Edition 3.0 edv aspekte (EDP Aspects) 3 (1984) p. 5.

Common Technology Programming

East Berlin EDV ASPEKTE in German No 3, 1984 (signed to press 30 May 1984) pp 56-59

[Article by Claus-Detlef Menschel, VEB ROBOTRON Center for Research and Technology]

[Text] The development of extensive and complicated software systems and their increasing number, the concomittant problems of planning, development efficiency, quality, and maintenance, intensify the requirements for making available industrial methods for the production of software products.

In the VEB Combine Robotron, ZFT Karl-Marx Stadt, the programming system "Uniform Technology" was developed. This programming system has been used for several years in a rather large development collective. Practicability, performance, and adaptability were demonstrated here.

The programming system Uniform Technology represents a computer support for all actions that are necessary in all phases of program development. It manages all data which accrue in these phases. It works under the control of the operating system SVM, Edition 3.0.

1. Introduction

Technological systems for software development have as their objective the comprehensive program support of all development phases of software products. For the practical use of technological systems, the following qualitative features have proven decisive:

- adaptability to specific user conditions and expandability
- dialogue capability with an effective response time behavior
- uncomplicated and always possible data access
- degree of automation of the formalizable processes, including the management of all data occurring during the development
- data protection
- reliability.

Among the operating systems made available within ESER, the operating system SVM, with its dialogue-capable programming and test system PTS, offers excellent presuppositions for the construction of a technological system. The operating system SVM is characterized by:

- efficiency of dialogue and batch operation
- the power of its editor, especially the editor XEDIT of the SVM Edition 3.0 /3/
- flexibility of data protection
- freedom of reorganization of the file system
- parallelism of execution processes

The program system Uniform Technology builds on these functions which are so essential for a technological system.

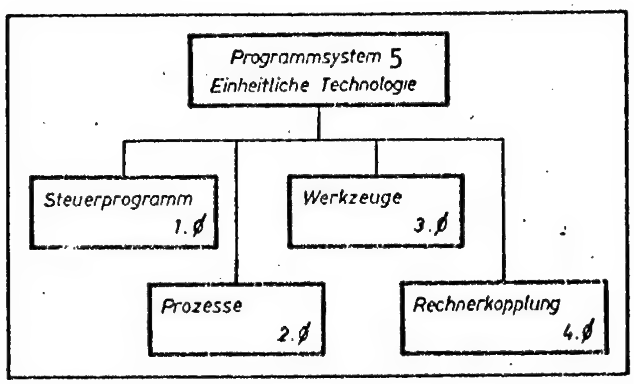
The objective of this program system is to take over the management of the data accruing during software development and to implement, in a standardized fashion, the processes that must be performed with these data.

The components of the program system Uniform Technology are shown in Figure 1.

2. Control Program

The control program of the programming system Uniform Technology works on a virtual machine (VM), which has available a virtual memory and a data space to manage the data and to execute processes. The virtual memory and the data space consist of one or more disks. The size of this space is determined by the quantity of data being managed. Only this virtual machine has write access to this data space. This machine will further on be designated as the central virtual machine (ZVM). All other VMs defined in the system have available only a read access.

Figure 1. Components of the programming system Uniform Technology



1 control program
2 processes
3 tools
4 computer coupling
5 program system Uniform Technology

By process is understood the processing of data with one or more programs.

2.1 Job and Process Management

All actions of the control program are based on so-called "jobs", which are transmitted by the user to the ZVM and which describe the data being handled and also define the processes and their sequences that are to be executed with these data.

At the same time, the resources required for each process are defined in the job.

The transfer of jobs from the user to the ZVM takes place through the SPOOL area of the SVM. The control program reads in these jobs, analyzes their syntax, and changes them according to the priorities assigned by the user.

When a job is selected, the processes defined therein are executed sequentially. The processing time of a process can be requested by the user within three different working regimes of the control program:

1. Execution should take place in an arbitrary time, depending exclusively on the priority of the job.

2. Execution should take place at a time when the overall load of the system has fallen below a defined threshold.

3. Execution in principle should take place only when a special processing time is defined by an operator command.

The objective of this definition is to achieve a uniform workload for the system and to displace load-intensive processes into the night-time where there is little dialogue operation.

By means of this measure, it is clearly fixed that in the standard regime 1 and 2 dialogue operation would have priority but in processing regime 3 batch operation would have priority.

If a process of a job cannot be executed at a certain time because of the working regime definition, the entire job is set back until the required working regime is reached and the next job is selected.

If a process is selected successfully, the control program checks whether the data media required for this process are available. Missing data media are requested automatically by the control program through a special operator interface. If these cannot be made available, the job is likewise set back.

The sequence of process executions can be controlled through conditional parameters, and an entire job can be interrupted under defined conditions.

Processes are connected to the control program through a standard interface. Since all processes are executed on the ZVM, they have read and write access to all data of the data space. For this reason, all processes intended for execution must be known to the control program by means of an agreement.

All programs of the processes, procedures, and control data are stored on data media, to which all - even the ZVM - have only read access.

Extensionsof the functional scope are thus possible only in a controlled fashion.

2.2 File Management

Within the framework of PTS, the file represents the smallest collection of information that is definable by a name and that is generated on the basis of defined rules. File management of the Uniform Technology programming system is designed in such a fashion that files can be handled which have arbitrary possible structures and formats within the PTS of the SVM. Thus, all the files (source texts, test data, test jobs, verbal documentation, etc.), which occur in the development of a software product, and which may differ as regards their content, format, and structure, can be managed.

The file management stores the files transmitted by the user within the data space of the DVM. Here, the file management itself represents a process in the job that has been transmitted by the user. Files with the same name are overwritten. Certain files that are essential for the organization of the programming system Uniform Technology can be protected by appropriate conventions against manipulations from "outside". The selection of the data medium for storing the files occurs exclusively on the basis of the criteria of uniform workload and is completely independent of the content of the files.

Since only the ZVM has write access to its data space, serialization of write operations, their checkability, and simultaneously the possibility of parallel reading without additional control mechanisms are automatically procured.

All storage actions undertaken by the file management are executed in compressed form. The elimination of blanks, in our developmental material, creates a gain of storage space by about 50 percent.

2.3 Protocols

The execution of jobs takes place in a time that differs a great deal from the moment of their transmittal and occurs completely independent of the particular user. For this reason, a protocol is kept on the ZVM, to make all activities and results visible. The protocol receives a message concerning each executed process of a job. This message contains the transmitter of the job, specified processes, and the return codes of the execution. Furthermore, statistical information concerning use frequency of individual processes and use times within the three working routines of the ZVM are kept in the protocol. The protocol is printed out daily. Safety copies are stored for three days in the ZVM.

If a job has finished executing, the job form, filled out with the return codes of the individual processes, is sent back to the VM of the sender.

For the individual user, as well as for the small group of persons that are required for monitoring purposes, adequate control possibilities thus exist for all activities.

2.4 Application Possibilities

The control program of the programming system Uniform Technology can be activated on several VMs simultaneously. These do not differ as regards their mode of operation, but can be set up for different objectives. For instance, they can manage different data or can implement different processes. In this way, greater parallelization and a better working mode of the system during night-time is achieved. The possibility of parallelization in our developmental material is used in such a way that a ZVM manages all development results and executes all the necessary processes with these data in a standard fashion. For partial collectives, so-called decentralized virtual machines (DVM) have been set up. These manage exclusively current developmental materials, that is data which are currently being processed and which do not yet represent a developmental result. The data space belonging to the DVMs is scaled accordingly to the tasks that must be implemented by the partial collective.

The user has read access to the data space of the ZVM and to the DVM belonging to his partial collective.

3. Processes

All operations which are implemented within the program system Uniform Technology by means of files furnished by the user, are implemented by processes.

Processes must be defined in the program system. They are called when needed through a standard interface, from the control program of the program system. Each process executes without user intervention. In this way, the user can be essentially liberated from taking notice of and controlling formal processes.

Table 1 shows some of the processes that are essential for our developmental material. At the same time, the corresponding use range within the programming system Uniform Technology as well as the developmental phase supported by this processing system is specified.

Table 1. Essential processes

Processes	Use Area ZVM	DMV	NVM	Developmental phase
STRUCDES	*	*	*	problem analysis, design, documentation
Macro management	*	*		implementation
Assembler	*	*	*	implementation
LINK	*	*	*	implementation
FINDSYM	*	*	*	implementation, documentation
FINDMAC	*	*	*	implementation, documentation
PTSUNLOAD	*	*	*	documentation (archiving)
TEPROS	*	*	*	documentation
Analysis processes	*			
SAVE	*	*		
STORE	*	*		

The listed processes are explained in more detail below.

STRUCDES

The program STRUCDES is used for the machine production of HIPO diagrams and supports the structured design on the basis of the HIPO method and the corresponding documentation /1/.

Macro Management/Assembler

Macros are managed in PTS macro libraries. If a macro is transferred for the first time or in modified fashion to the ZVM, the corresponding library is updated through the macro management.

The macro management makes sure that the current editional macro states are used in principle during translations, without being influenced by the user. This leads to a significant reduction in the number of errors.

LINK

This execution process builds up a job to execute the program linkage, so as to produce a load module under the control of an OS/ES operating system. The target library and the necessary parameters for the program linker are taken from the source text of the corresponding module. The results of the link run are analyzed by the processor.

FINDSYM

The most economical error search is the dry test. The program FINDSYM works under the control of PTS and determines the use of specific symbols or symbol groups for requested source texts, which are written in Assembler code. A well-aimed dry test is possible by means of this information.

Special symbol refernece lists, that are suitable for program documentation, can also be produced with FINDSYM.

FINDMAC

The program FINDMAC works under the control of PTS and determines the use of macros, including internal macros, for requested source texts that are written in Assembler code.

PTS UNLOD

The program PTS UNLOD works under the control of PTS. It copies specific data on tape in compressed or decompressed form. The structure of the resulting tape files is appropriate for OS/ES.

The program is essentially used to file developmental statuses.

TEPROS

TEPROS is a text processing system /2/ which works on a VM under the control of SVM.

TEPROS is used to produce verbal documentation.

Analysis Processes

Various analysis programs are implemented in the programming system Uniform Technology. These implement the following tasks:

- testing the files transmitted by the user for formal correctness
- producing and maintaining a change documentation including print preparation
- keeping quantitative statistics as regards the scope of development and the use of computer time per user.

The analysis programs promote the quality of the developmental results. At the same time, they make available easily callable specific data at any time

to the manager of software development collectives. This facilitates estimation of the achieved developmental status and makes it possible to analyze the overall developmental process. These data create the necessary preconditions for continuous methodical work.

SAVE

The execution process SAVE implements data saving. Arbitrary disks can be saved with the execution process SAVE.

Up to now, the data media were saved completely. Within the framework of further development, it is intended to save only the files that have changed with respect to the last saved status.

STORE

The execution process STORE implements the storage of a file transmitted by the user in the data space of the ZVM.

4. Tools

Data manipulation is performed by the user on a virtual machine that is proper to the user (NVM). Here, when the programming system Uniform Technology is used, the multiplicity of possibilities which the user working on a VM possesses through the simulation of the real system and through the broad functional scope of the VM is in no way restricted. The PTS editor, the PTS functions for file processing, the EXEC processor of the SVM, and the CP capabilities thus have remained the essential tools for data manipulation and testing. Building on the capabilities of the PTS, specific tools (commands and procedures) were furthermore created for standard problems. Since no user has write access to the files managed by the ZVM and DVM, and the searching and procuring of files by the user himself would be too laborious, although in principle possible, procedures are furnished, among others, to execute the following functions:

- transfer of files to the ZVM or DVM
- the furnishing of files from the ZVM and to the NVM
- renaming of files
- erasing of files
- printing of files
- generation of files.

All these procedures represent corresponding jobs, which are transmitted to the ZVM or DVM.

By using the structure of the file designation that is usual in the PTS

filename.filetype

the content of the file is characterized in the filetype, by means of an appropriate convention which holds for all staff members participating in

the development. In order to facilitate the simultaneous development and maintenance of several program versions, this designation also receives a program version number, for example SYM619 for a source text of version 619 and MAC700 for macros of version 700.

Starting from this information, the procedures are capable of generating standard executions of various processes for one job. At the same time, the appropriate interrupt conditions and the working regime of processes as well as the priority of the job are generated by these procedures. For example, if a source text file is transmitted to the ZVM, the transmission procedure generates the following process sequence:

- process to test the job
- data analysis process
- store
- furnishing the macro library corresponding to the specified version
- assembler
- LINK

The user can easily vary these generated jobs by way of parameters, but also by directly manipulating the job file itself.

Besides these procedures, there are commands by means of which the working regime of the control program as well as job priorities can be changed and by means of which stop and termination conditions can be set.

5. Computer Coupling

A data space of a ZVM or DVM can be managed only at one computer. When working with several computers, a meaningful subdivision of the DVMs and the associated subdivisions of the development collective to the corresponding computers is necessary. Nevertheless, at least access to the data of the ZVM is necessary from all computers. The computer coupling between two computers is implemented by an adapter, which consists of two VMs, the AVM1 and the AVM2, each with a data space. For this a presupposition is a shared DASD connection of the corresponding WPS. Both AVMs have write access to their own data space and read access to the data space of the other AVM. The AVM 1 works under the control of the SVM of computer 1; the AVM 2 works at computer 2. A control program is always active on both AVMs. It checks whether requests exist on the partner AVM. If yes, the requested data are procured, are stored in their own data space, and from there are furnished to the user by the other AVM, or to the ZVM or DVM.

The solution explained here permits access to all data at all computers by all users. To improve the response time behavior, the use of computer coupling via KKA should be striven for.

6. Expandability and Capability for Adaptation

The development of the programming system Uniform Technology took place for a specific developmental subject and the resulting file types, processes, and design and documentation methods.

The implementation of generally valid technological systems can be implemented only on the basis of expandability and thus on the basis of adaptive capability to concrete user needs. The programming system Uniform Technology guarantees the following expansions:

- arbitrary PTS-supported data formats and data structures
- arbitrary programs, which can function within PTS as a process, among these primarily all compilers that can run within PTS
- arbitrary tools that can be produced by means of PTS-EXECs or programs

Thus the programming system Uniform Technology has high user adaptability.

After comprehensive adaptation of the programming system, the following improvements can be claimed:

- quality improvements by uniform and automatic tests, perceptible reduction in the number of error sources on the basis of standardized processing
- secured response time behavior in dialogue operation by providing relief for peak-load times
- faster reaction to necessary program changes through the significantly higher availability of data
- furnishing constantly updated test lists and protocols to control the state of developmental work
- automatic data protection
- increase of productivity and saving of computer time through the automation of formal work and the relief of staff members in research and development.

References

/1/ Menschel, C.D.; Neumann, J.: Program support STRUCDES for structured design. edv aspekte (EDP Aspects) 3 (1984) 3, p. 60.

/2/ User documentation TEPROS. VEB Management Center for Applications Research.

/3/ Heinecke, K.; Schroeder, A.: System of virtual machines - SVM, Edition 3.0, edv aspekte (EDP Aspects) 3 (1984) p. 5.

8348
CSO: 8120/0248

GERMAN DEMOCRATIC REPUBLIC

STEPS TOWARD FACTORY AUTOMATION IN WORKPIECE PRODUCTION

East Berlin NEUES DEUTSCHLAND in German 20 Nov 84 p 3

[Article by Heinz Singer]

[Excerpts] Manufacturers of NC processing centers, complicated production systems, and special machinery, have to offer their clients, in order to be able to compete in the international markets, constantly novelties offering obvious advantages in production and productivity. This has been possible, however, in the line only by the most rationalized combination of the machine building technology with modern electronics.

Precisely that has been mastered already for years successfully by the mechanical engineers in Saalfeld. None of the presently manufactured products are in production more than three and a half years and all are bearing the quality mark "Q". In 1984 the renewal rate will even exceed 40 percent. The 1720 machine builders want to keep this up also in 1985, increase the production by at least ten percent, and thereby preserve the chance to have their products in the markets always at the right moment.

Such a response capability on the part of the Saalfelders goes hand in hand with far-reaching considerations of the production method. These considerations meet to a far-reaching degree the economic strategy decided by the 10th party conference of the SED. Plant manager Fritz Schleizer: "The response capability per se is not what counts in business management and, of course, in the national economy, but most of all how it is realized in technical respects and thus mastered economically. Only then becomes the intensification complex and profound. Our concept is therefore the manufacture of top-notch products at the lowest possible cost."

Tailored to the Demands of the Times

How both are to be tackled is demonstrated today by the Saalfelders in one of their oldest factory buildings, where the first steps towards an automated manufacturing plant for the pre-production of workpieces are becoming technical reality. Recently, a completely automated production section has started production in three shifts on small and medium series of 700 different axially symmetrical parts. Five technological units -- among them numerically

controlled machine tools as well as robots -- are linked to transfer stations and pallet storage shelves. All work processes, including transport and storage, will be completely automated. At the present time, the Saalfelders are starting operation of the microelectronic control of the shelf-operating equipment. The passage of the parts will then take only half as much time as previously and the productivity will rise by 60 percent. 18 workmates will be freed for other tasks and strengthen the work potential in other areas.

This is a scientific-technical top performance serving as an example for the machine construction in the GDR, especially since the Saalfelders have accomplishedit by their own efforts. Engineer Hors Mueller, group leader for rationalization and with the enterprise for 24 years, could thereby, like all workmates, link up with experiences gathered in Saalfeld already in the past when steps were taken into unknown territory. At present, he exchange his work place with the lecture hall at the Technical College in Karl-Marx-Stadt and participates in a three weeks course of further education. Not for the first time, he says, for qualification was an obligation for all researchers, designers and technologists so that one can perform one's daily duty. "That way one can cope with all problems arising on the way into unexplored territory. Without this way, however, a comprehensive intensification cannot be realized. We achieve the increase in output we need for the further shaping of the developed socialist society."

Built in the Own Workshop

A germ cell for the new in Saalfeld -- this becomes clear from the words of Horst Mueller -- is a solid work of persuasion both from a political and tehcnical aspect. The designers and technologists of the automated plant appeared in party and trade union collectives, outlined their ideas on the blackboard, explained their intentions on a model, and illuminated thereby correlations for the Saalfeld contribution to the successful continuation of the politics of the main objective. Popular-scientific lectures and many activities of the KDT section of the plant, which like the FDJ assumed numerous partial work tasks, contributed to winning the entire plant collective for the courageous plan.

Top-notch products, first rate technology -- the Saalfelders add a third link in the chain to the first two: Top technology is created efficiently in the own workshop. Socialist community spirit makes it possible to plan and build directly articulated robots and automation plants for transport and storage with the use of microelectronic controls. "Equipment manufactured for own requirements constitutes at our plant for years between four and five percent of the total production", comments test engineer Willi Salmann who participated from the start in the assembly and testing of the automated facilities. "But never -- hats off to our designers -- were so many and novel ideas born as in this project. And my respect to the partially still very young colleagues at the center of the efficiency structure who converted what was drafted on the drawing-board to practical application." In one of the oldest factory buildings despite a lot of skepticism in the course of the production run.

Everything described in the foregoing would have remained incomplete in the end if the Saalfelders had not turned into reality at the same time jointly with the data processing center of the city a further project from the state plan for science and technology dealing with remote data processing the importance of which was explained to us by the plant manager Fritz Schleitzer, as follows: "Our production is comprised of about 18,000 operating cycles daily. In order to have every part reach its predetermined destination at the desired time, we have now stationed display units and printers in all craft areas, which are connected by way of a long-distance line with the large computer in the Saalfeld plant of the VEB Data Processing Center Gera and which receive the optimal replies from there."

Computer Station with 40 Experts Saved

Here too, one cannot fail to note the manner in which the Saalfelders run their economy and policy. Only the storage capacity of the large computer had to be expanded. They were successful in converting expensive routine areas in the planning and control of the entire reproduction process of an industrial plant to electronic data processing and computing operations. And in contrast to known solutions in enterprises with their own computer station the Saalfelders could dispense with investments of about M 6.8 million. At the same time, about 40 EDP experts would have been required for manning an own computer station.

Let us return to the point of departure. There are therefore important reasons for going in the direction of the automated factory. A new plan has therefore been announced in Saalfeld: The automated production of portable drills up to their packing.

12693
CSO: 2302/53

GERMAN DEMOCRATIC REPUBLIC

BRIEFS

MINIATURIZATION OF RELAYS IMPROVED--Imagine taking 2,000 relays in hand daily and testing and sorting them four times each. The exertions of this manual activity become especially clear when products of the VEB Relay Engineering Grossbreitenbach, such as the recent new development from the original roughly matchbox size are "shrinking" now to the by a multiple smaller size of a microprocessor circuit. Each of these parts miniaturized for the requirements of microelectronics must still prove, however, its functional safety prior to its delivery to the user. In order to automate such processes right from the start of serial production of new products, the enterprise has taken for some years now successfully the road of cooperation in science with the Technical University in Ilmenau. Since the beginning of the year, nine robot processes were made operative for practical application essentially on this basis. The VEB Relay Technology is connected for this purpose by special complex agreements with the technical university in the same way as 14 other combines and enterprises. This contract, explains Rudi Boenisch, director for science and technology, contains long-term projects for which both facilities will work out jointly specifications. Moreover, the Technical University offers the plant the opportunity to delegate engineers for certain periods in form of special planning locations, for instance to its appliance engineering section. It had proven worthwhile that the specialists working there solved scientific-technical tasks up to and including research projects, following which they returned to the plant with these developments in the study of a functional sample. The latter are developed thereafter to maturity for serial production. "We have ascertained to date that this kind of cooperation is promoting especially the creativity of our engineers," says the director. Thus, the enterprise fulfills, among other things, its plan of producing new products from the start with new technologies. This applies also to the new relays which require only one-tenth of the expenditure in materials as comparable products and can be used as bulk materials in the automatic assembly of printed circuit boards. They are tested thereafter by a freely programmable, computer-based automation equipped with data storage and are then sorted according to their specific application. In addition, the automaton combines four former operations in one process. The engineers availed themselves hereby also of experiences gathered in earlier solutions, with potential increases in production of up to 700 percent. This development is to be continued in 1985 with the realization of seven additional automation subjects. /Text/ /East Berlin NEUE ZEIT in German 30 Oct 84 p 6/ 12693

LASER SEPARATION FOR IC PRODUCTION--A technology for separating oxideceramic carrier material for a group of microelectronic hybrid circuits was developed in the Ceramic Works Combine, Hermsdorf. The heart of the equipment is a laser unit from the Precision Mechanical Works Combine: Halle. The ceramic sheets manufactured in a size of 65 x 105 mm and only slightly thicker than 0.5 mm can be divided into smaller sections with the new technology. The sheets are moved with computer-control under the laser beam which applies up to one thousand extremely fine melting points per second to their surface. The carrier material can thus be broken without any problems along the scoring and thus be divided with high precision into sections of the desired dimensions. /Excerpt/ /East Berlin NEUE ZEIT in German 22 Nov 84 p 1/ 12693

ANNUAL STEEL PRODUCTION FIGURES--A look at the converter steel works in Eisenhuettenstadt. It conveys an impression of the size of the plant. The entire process at the converter, from the preparation of the melting charge, through the blasting process up to the tapping is monitored by a computer-controlled process from the central control station. While one converter is at work--210 tons of steel are produced in 38 minutes--the other is fitted with a new refractory lining and is ready again for operation at the end of the production cycle of the first converter. Two million two hundred thousand tons of raw steel and 2.1 million tons of semifinished products can be produced here annually. [Text] [Photo Caption] [Frankfurt/Oder NEUER TAG in German 22 Nov 84 p 3] 12693

MACHINE TOOL MODERNIZATION PLANS--The Berlin parent plant of the Machine Tool Combine "7th of October" is equipping its products increasingly with microelectronic controls. It is planned for 1985 to manufacture all machines of a new series of tooth profile grinders with such controls from the GDR production. The shop stewards of the parent enterprise consulted at the plenary meeting on Tuesday on this and other competitive projects for the coming year. They discussed jointly with the minister for machine tool and processing-machine construction, Dr Georgi, how an even greater contribution could be made by machine tools and technological solutions with integrated robotics for further rationalization in the metal working industry. The Berlin machine tool builders want to develop and produce in 1985 additionally a technological unit for the machining of dynamically balanced parts. Ninety-four percent of the planned increase in output are to be brought about by scientific-technical performances. The productive capacity in the area of product development is to be increased further with the aid of the scientific organization of work, explains the Director General Dr Heinz Warzecha. The renewal rate of the entire range of products will be increased by more than one-third. The complex rationalization of entire production sections shall make it possible to save more than 60,000 work hours. The working conditions of 180 work places will be improved in 1985 primarily by the employment of robotics. The trade union representatives resolved to surpass the plans for net production, production of goods, work productivity and the production of finished products for the population by 1.2 percent each. /Text/ /East Berlin NEUES DEUTSCHLAND in German 28 Nov 84 p 4/ 12693

COMBINE MODERNIZING 1,500 MACHINE TOOLS--The machine factory Meuselwitz has assumed complete modernization of machine tools manufactured by it in the past and will deliver for this purpose also the necessary documentation. This year

high-quality machine tools were rebuilt according to the latest findings, among others in the roller foundry Coswig, in the special steel plant Lugau, and in the VEB Plamag Plauen. The output of roller grinders rose thereby by at least 25 percent, while energy consumption declined by 12 percent. "Engineers, technologists, and all participating collectives devoted all their energy to an acceleration of the modernization program. It is clear to everyone: That which strengthens the GDR will make peace more secure," said plant manager Dr Guenther Rippin. In 1984 the machine tool combines will already modernize about 1,500 machine tools. The production of technology for the modernization of machinery is to be expanded considerably during the coming years. While the machine tool combines are delivering this year structural components for modernization in an amount of M 90 million, this volume is to be increased in 1985 by 50 percent. /Text/ /East Berlin NEUES DEUTSCHLAND in German 22 Nov 84 p 1/ 12693

INDUSTRIAL FACILITY CONSTRUCTION COMBINE--Facts and figures: The VEB Central Facility Construction Combine of Metallurgy (ZIM) is in existence since 1 July 1984. The parent plant is the VEB Automation Berlin. Among the tasks of the combine are the production and employment of specific and flexible robotics and microelectronic control mechanisms in technological units, as well as the new construction and redesigning of complete metallurgical thermic and furnace facilities. ZIM is one of the few enterprises in the GDR manufacturing robots in larger series for a wide circle of users. It has delivered 460 robots since 1980, among them the ZIM 60, its smaller brother ZIM 10, and since recently the ZIM-10-1. It is envisioned to build 460 robots in 1985, about 40 of them as compelte installations, for instance for welding. These flexible, articulated robots are intended primarily for the metallurgy and mining industries, as well as the central rationalization structure. The FDJ initiative "industrial Robots" has become a major proving ground for young workers and young members of the scientific-technical intelligentia. Its objective is the production of 10,000 robots by the end of 1985. /Excerpts/ /East Berlin WOCHENPOST in German No 45, 9 Nov 84 pp 4-5/ 12693

CSO: 2302/53

YUGOSLAVIA

BRIEFS

DOMESTIC ROBOT--Trstenik, 7 January (TANJUG)--The Prva Petroletka plant in Trstenik, in cooperation with the Mihajlo Pupin Institute in Belgrade, has produced the first domestic universal manipulation system--UMS-3. At the present time this first Yugoslav robot is in operation at the Sloboda plant in Cacak, where it controls a 630-ton press. [Text] [Ljubljana DELO in Slovene 8 Jan 85 p 2]

CSO: 2802/1

- END -